An Automated Internet-Based Robot Soccer System

LONG, Qiaoxi

A Thesis Submitted to the Chinese University of Hong Kong in Partial Fulfillment of the Requirements for the Degree of

Master of Philosophy

in

Mechanical and Automation Engineering

The Chinese University of Hong Kong

May 2011

Thesis/Assessment Committee

Professor Chung, Chi-kit Ronald (Chair)

Professor Liu, Yun-hui (Thesis Supervisor)

Professor Wang, Changling Charlie (Committee Member)

Professor Ning Xi (External Examiner)

Abstract

A fully automatic internet-based robot soccer system is described in this paper. Because the existing robot soccer system is not intelligent and integrated enough, the popularization of such system is restrained. Meanwhile, human maintenance workload is also very heavy. This proposed system applies several mechanisms such as auto-charging, auto-scoring and auto-judging which enable the games to run automatically without manual interventions. Internet accessibility is also integrated in this system which allows the users to play the game remotely.

In auto-charging, computer vision is used to detect the locations and orientations of robots on the playground. Power monitoring module detects the battery level and report to the server using RF communication module at certain interval. Motion control of differential mobile robots drives the robots to desired positions, such as the charging chambers for the low-power robots. The control instructions are sent in a format of left and right wheel velocities (PWM value in our case) by RF module. For auto-scoring and auto-judging, an expert system is built to define the game rules. A special judge robot is made to move the ball automatically when needed. Besides, a website is also built for players to join the game remotely and manage their game information.

Finally, performance of the proposed system is illustrated by the trial running. The result shows that the robot soccer system is strong enough to conduct an automatic internet-based robot soccer competition without any manual interventions.

摘要

這篇論文介紹了一個基於與聯網的全自動機器人足球系統。由於現有的機器人足球系統還不夠智慧化和集成化，該系統的推廣因此受到很大制約。與此同時，它需要的人工維護也非常繁瑣。我們在此提出一個新的系統。它應用了諸如自動充電，自動計分和自動裁判等機制，使比賽能夠在沒有人工干預的情況下自動地進行。互聯網的可訪問性也被集成於該系統中，使得遠端的遊戲者也可以參與比賽。

在自動充電機制中，機器視覺被用來檢測場上機器人的位置和方向。電源監測系統通過無線射頻通訊模組將它監測到的電量資訊以一定的間隔不斷發送給伺服器。差分機器人的運動控制系統被用來將機器人驅動到一個特定的位置和角度，比如當某個機器人電量低時，該控制器將該機器人驅動到我們特製的充電槽中進行充電。控制資訊是以左右輪速度的 PWM 值發送到每個機器人的。我們為自動計分和自動裁判機制建立了一個專家系統，用以定義比賽的規則。同時，我們還設計了一個裁判機器人，用來在必要的時候將球移動到特定的地點。除此之外，我們還建立了一個網站，遊戲者可以在那裡進行遠端遊戲或者管理自己的比賽資訊。

我們在試運行中證明了該系統的性能。測試結果顯示該系統能在沒有任何人工干預的情況下進行基於互聯網的全自動的機器人足球賽。

Table of Contents

Abstract 2

摘要 3

Table of Contents 4

Acknowledgements 6

List of Figures 7

List of Tables 9

Chapter 1 Introduction 10

1.1 Robot Soccer 10

1.2 IRIP and IRIS 11

1.3 Motivation and Literature Review 14

1.4 Technical issues and Contributions 15

1.5 Thesis Outline 16

Chapter 2 The IRIS system 17

2.1 Hardware setup 17

2.2 Software architecture 21

Chapter 3 Internet Accessibility 26

Chapter 4 Auto-charging 32

4.1 Hardware setup 33

4.2 Communication 36

4.3 Vision 41

4.4 Motion control 44

4.4.1 APF 44

4.4.2 CRC 50

4.5 Processing Schemes 51

Chapter 5 Auto-Scoring and Auto-Judging 56

5.1 Auto-scoring 56
5.2 Auto-judging 57
5.3 Judge robot 59
Chapter 6 Experimental Results 63
Chapter 7 Conclusions 69
7.1 Summary 69
7.2 Future work 70
Appendix A 72
Bibliography 73

Acknowledgements

Firstly, I would like to thank my supervisor Prof. Yun-hui Liu for his support since 2007. He is such a wise man that you can always learn a lot from him. His encouragements and patience are so appreciated. I would also like to thank my parents who has supported and guided me for over 24 years. Their unconditional love is the most precious treasure in my life.

Members of IRIP group, it is so lucky to have you in the same group for over 2 years. Martin, Wang Wei, Tai, Ding Ke, Qing-yun, CT, and Frankie, thank you very much for your help. Without your help, I would never able to have this thesis. I would like to thank all other members of NSRL. Work with your everyday is one of the happiest things indeed.

List of Figures

Figure 1.1 General setup of a FIRA game 11

Figure 1.2 Game field, robots and the ball of IRIS 12

Figure 1.3 Online-design system (The assembling of a car) 13

Figure 1.4 Robots representing Hong Kong, Beijing and Macau respectively . 13

Figure 2.1 Hardware architecture of IRIS 17

Figure 2.2 Differential Drive kinematics 20

Figure 2.3 Overview of software modules of IRIS 21

Figure 2.4 Traditional client interface of IRIS 23

Figure 3.1 Array transmitted between server and client 27

Figure 3.2 The flash program for IRIS 30

Figure 3.3 User interface with videos of players and game field 31

Figure 4.1 Overall flowchart of the automated IRIS system 33

Figure 4.2 New game field built for Automated IRIS and chamber numbers... 33

Figure 4.3 The charging chamber 34

Figure 4.4 The robot for new system 35

Figure 4.5 Robots inside the charging chambers 35

Figure 4.6 Co-occurrences puzzle 42

Figure 4.7 Flowchart of vision module in Auto-Charging 43

Figure 4.8 Position and orientation of robot w.r.t the target 46

Figure 4.9 Attractive potential field 47

Figure 4.10 Repulsive potential field 48

Figure 4.11 Combination of artificial potential fields 49

Figure 4.12 Narrow gaps when robot entering the chamber 50

Figure 4.13 Functional modules of IRIS server 52

Figure 4.14 Robot Blue 1 right in front of the chamber 1 52

Figure 4.15 Two steps to drive a robot into the charging chamber 54

Figure 4.16 Initial positions of the robots 55

Figure 5.1 Architecture of a typical expert system 57
Figure 5.2 Particular zones and points on the game field 58
Figure 5.3 Judge robot catches and releases the ball 60
Figure 5.4 Color patches of judge robot 61
Figure 6.1 Procedure of changing one robot 64
Figure 6.2 Procedure of changing six robot 66
Figure 6.3 Processing procedure after a goal 67
Figure 6.4 Processing procedure after a goal 68
Figure A.1 The playground dimensions of IRIS 72

List of Tables

Table 2.1 Color patches of different robots 19

Table 2.2 Bytes sent by RF emitter (former IRIS version) 24

Table 2.3 Velocity values sent to RF emitter 25

Table 3.1 Position and orientation values sent by server 27

Table 3.2 Relationship between variable *i* and the corresponding objects 27

Table 3.3 Status messages indicated by Array [0] 28

Table 3.4 Elements in Array denoting scores and time 28

Table 3.5 Array sent by clients to the server 29

Table 3.7 Numbers of robots 29

Table 4.1 ID of Robots 36

Table 4.2 Command packet sent to ACS by server 37

Table 4.3 Action command list (server to ACS) 37

Table 4.3 Response packet (ACS to server) 38

Table 4.4 System status list 38

Table 4.5 Error report list 39

Table 4.6 Command packet from Server to RF emitter (after extension) 40

Table 4.7 Corresponding robots of array Status 42

Table 4.8 Attractive forces along x and y axis 46

Table 4.9 Repulsive forces along x and y axis 47

Table 4.10 Chamber number for each robot 53

Table 4.11 Groups when changing all six robots 55

Table 5.1 Typical rules in IRIS 58

Table 5.2 A byte controlling the gripper of robot soccer 60

Chapter 1 Introduction

1.1 Robot Soccer

There are two most popular robot soccer tournaments now in the world: FIRA and Robocup, which are also the founders of robot soccer. FIRA, Federation of International Robot-soccer Association in full, was initiated by Jong-Hwan Kim, KAIST, Korea in 1995, and the first tournament was held in 1996. Robocup can be dated back to a workshop on Grand Challenges in Artificial Intelligence organized by a group of Japanese researchers in October 1992 in Tokyo. The first official RoboCup games and conference were held in 1997.

Both FIRA and Robocup contests include several leagues which are different in size, complexity, locomotion, and rules. In Robocup, it includes simulation league, small size league and humanoid group etc. Correspondingly, FIRA also has SimuroSot, MiroSot and HuroSot, etc. Take FIRA for example: SimuroSot contests are usually played online with two clients connected to a simulation server. Players can define strategies on the client ends and monitor the realtime simulated games. It mainly focuses on artificial intelligence and educational purposes. MiroSot and HuroSot contests are played by mobile robots or humanoid robots. It requires more technologies including robotics, mechanism, wireless communications, etc. Here, as our IRIS system is similar to the MiroSot (small size league in Robocup), we put our focus mainly on it in this article. The MiroSot is played by two teams, each team consisting of 3 mobile robots (one of which is the goalkeeper). A host PC is provided to each team for image processing and location identifying. MiroSot requires the players to define good strategies and develop sharp sensing as the competition is so comlicated. The control commands are generated according to the realtime situation of the game and then sent to the robots by wireless communication system (eg. IR or RF).

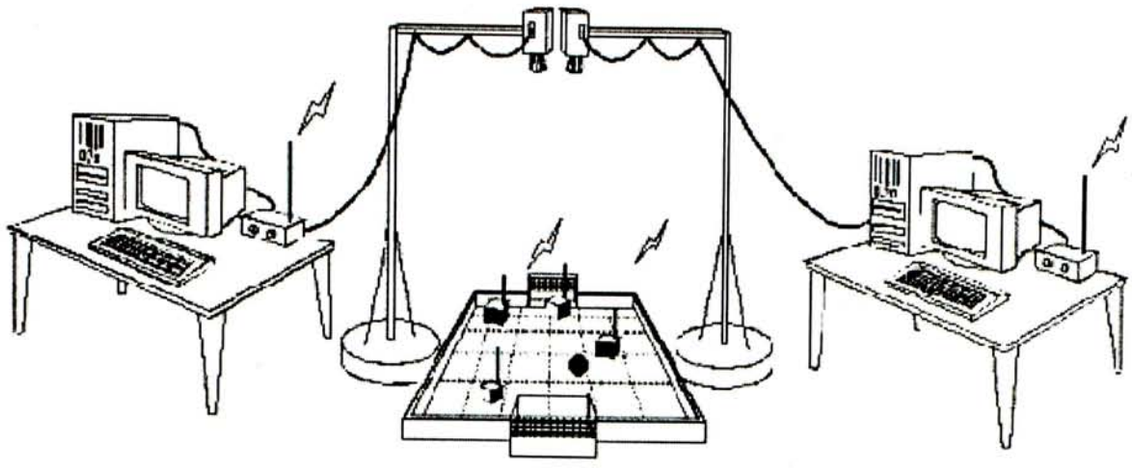


Figure 1.1 General setup of a FIRA game [1]

1.2 IRIP and IRIS

Internet Robotics Innovation Park (IRIP) is a project hosted by Chinese University of Hong Kong. It is granted by ITF (Innovation and Technology Fund of Hong Kong). As a platform to promote technology awareness and innovation for the general public, especially the youth, it is a place where people can share ideas, inspire innovations, demonstrate and share the design work of robots, learn, design, verify their robots through simulation tools and so on. It contains four main elements: competition, robot design, exhibition and education.

Competition is a most important element in IRIP as it always can generate the enthusiasm of the players. Competitions of IRIP includes Internet Robot Inter-School Competition (IRIS), Homanoid Robot Dance Competition, Lego Robot Competition as well as Service Robot Competition etc. Nowadays, IRIS is the main character in IRIP. It has been holding for several years and attracts many people, especially the students. It is a robot soccer game played by 6 players, three players on each side. The structure is very similar with that of MiroSot in FIRA. In IRIS, players use joysticks

linked to their client PC to control the robots. Game field, robots and the ball of IRIS is shown in Figure 1.2. IRIS program provided an opportunity to the public to take part in the soccer games which would inspire their creativity and enthusiasm for robotics. Details of IRIS are to be introduced in the next chapter. Beside the IRIS, other forms of IRIP contests are to be developed in the near future.

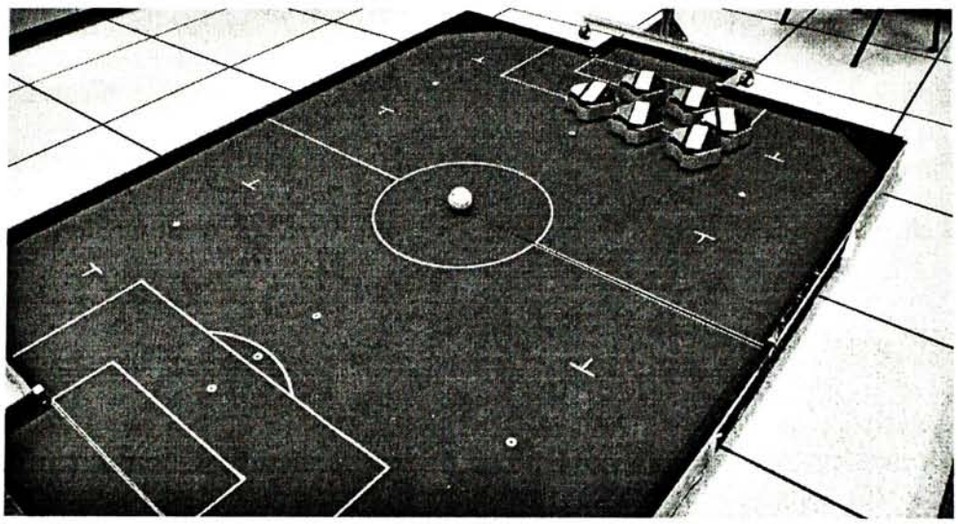


Figure 1.2 Game field, robots and the ball of IRIS

Robot design includes Online Robot Design and Sharing of Design Works. Players can open a flash program on our website and do the design online. Such function is still under construction. Now, as a beginning, we provide some components such as gears and shafts which enables the players to assemble a car (see Figure 1.3). Currently, the competitors can design their own robots in real world and share their works with others every year during competition (Figure 1.4).

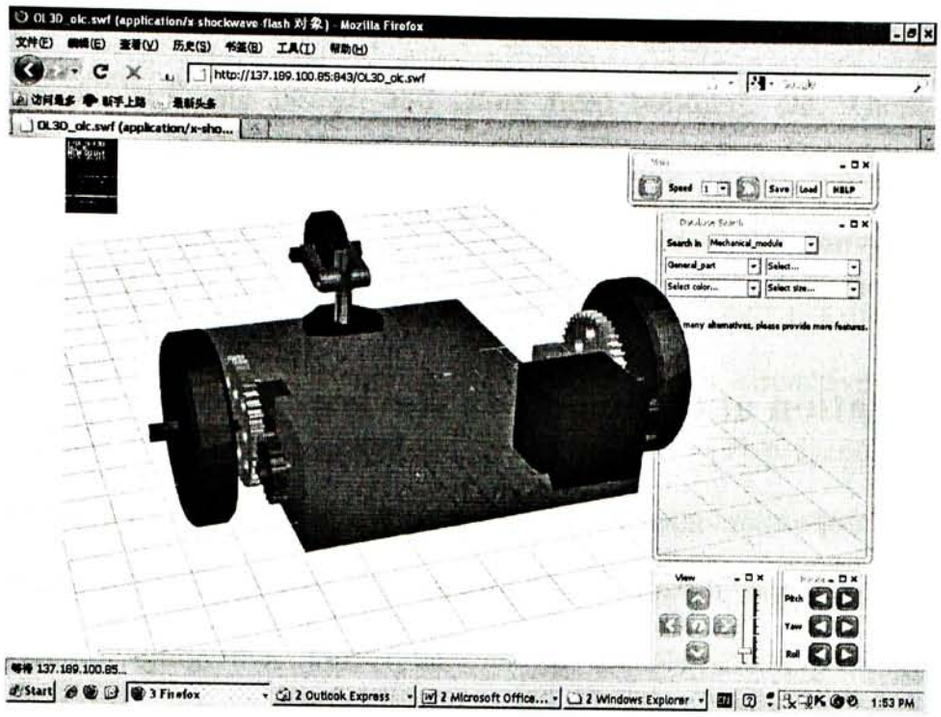


Figure 1.3 Online-design system (The assembling of a car)

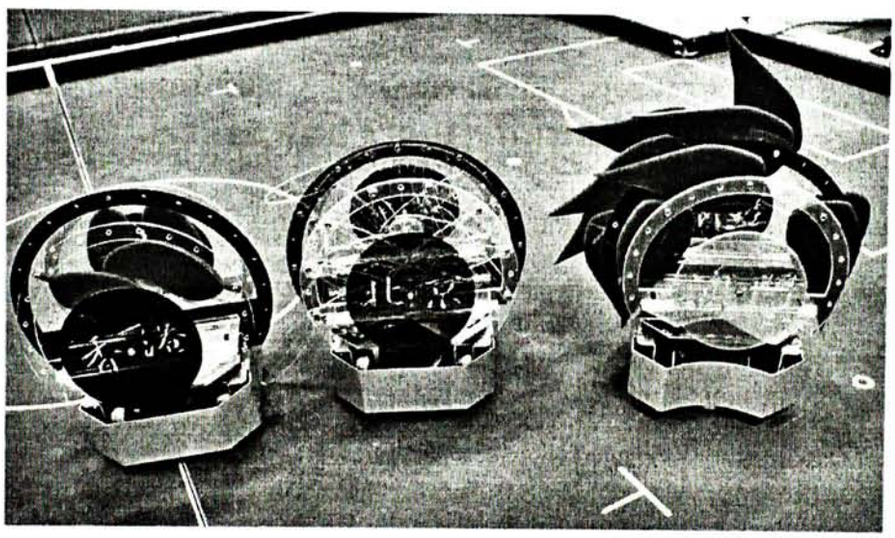


Figure 1.4 Robots representing Hong Kong, Beijing and Macau respectively

We have some documents and workshops related to robotics on our website. We also organize some summer or winter camps for the youth for their better understandings to robots. We will also have Online Labs in future. In all, education meaning is the one we will always consider first.

We have also a legend that, one day, we would have a Worldwide Robot Research Exhibition in IRIP. People can share their newest design online or offline. Customers

from all over the world would be enabled to access the online purchase system for their preferred products. This is the fourth component of IRIP: the exhibition function.

1.3 Motivation and Literature Review

Robot soccer competition is now very popular world widely because of its strong amusing recreation and instructive significance especially for the youth. It also attracts many researchers because it contains various emerging subjects such as robotics, sensor fusion, intelligent control, communication, image processing, mechatronics, computer technology, artificial intelligence, etc[1].

The existing robot soccer competitions are always controlled by the judge. Take FIRA for example: judges' duties include changing batteries, scoring, judging and positioning of the ball and robots, timing, etc. Judges must suspend the games and substitute the robots or changing batteries manually [2]. Such interruption will surely affect the visual quality of the games. The judging of the FIRA is always conducted manually. However, due to the limitation of human vision, players always had altercations over the judge's decisions. The workload is heavy, but the result is not usually good. The whole game course is controlled by the judge, which means that if there is no judge, the game cannot be conducted. However, if we want to promote the robot soccer to the world, we must enable players from any corner of the earth play the game at any time. However, it is not very possible for us to have a judge at any time. What else, the existing robot soccer systems, including FIRA and Robocup, are mostly C/S based. It means that such system can only be used locally. This would cause some problems to those remote players since they may have difficulties to come to the venue. It will bring some negative influence to the popularity of the robot soccer.

In robot soccer system, most of the research issues focus on the robots, such as the motion control of mobile robots and the vision part. Few literatures are about the systematical design of an automatic system. However, we can divide the

[1] See http://www.fira.net or http://www.robocup.org for an overview

complicated problem into several parts, such as motion control of mobile robots, computer vision, hardware design and game field building, etc. There are many literatures about motion control of mobile robots. Gyula Mester introduced a fuzzy reactive controller of a mobile robot motion in an unknown environment with obstacles [3]. R. Fierro and F. L. Lewis developed a neural network-based controller designed for motion control of a mobile [4]. [5] introduced a reactive approach that makes use of the Vector Field Histogram (VFH). [6] and [7] presented two motion control methods in which only first order of kinematics are considered. For the vision part, [8]presents a system that can adapt to rapidly varying light and coloring conditions, while maintaining speed and accuracy. [9] presents a flexible and robust vision system for robot soccer. It is easy to use and to adapt and delivers stable results. A global vision scheme for estimation of positions and orientations of robots is presented in [10]. [11] also gives an example of the hardware implementation of a micro-robot soccer team.

1.4 Technical issues and Contributions

There are many technical issues to be discussed:

- To do the auto-charging, our idea is to have two groups of robots. Each robot has its brother in the other group. When one is playing, its brother is charging in the chamber. When the one on the field goes out of battery, we exchange them. But how to identify the robots on the field as there may be two same robots on the field at the same time. It may cause chaos in the current system.
- Due to the limitation of cost, we failed to use step motor on our robots thus caused some inaccuracy. But, how to drive the robots into the charging chamber fast and accurately as the chambers is narrow compared to the width of the robots.
- How to know whether the battery level of a robot is low? We can only judge according to their moving speed which is not accurate at all.
- How could the computer do judging and scoring automatically instead of

human beings?

- How to enable students outside to play our games and how to improve their experience?

After some hard works, we finally finished a first version of the proposed new system. The contributions of such system may be:

- Made the system more robust by developing a new computer vision module which can indentify two groups of robots as the robots need to be changed when out of battery.
- Added more functions by developing robot charging system which can do the homing and charging of the robots automatically. At the same time, we designed new robots as to do the auto-charging and new control methods which were used for better convergence of motion. The motion trajectory is more smooth and shorter. It would cost less time and promote efficiency.
- Auto-judging is done by designing a new judge robot and a rule set. It improved the robustness of the robot soccer system even more as the scoring and judging can be conducted by this module automatically. No more human judges were needed and therefore reduced the expense of the contests.
- A flash program is uploaded to our website to conduct the online games, which can improve the user sentiments.

The result system can completely meet the design requirements: It can conduct a fully automatic online robot soccer game. Simulations are presented at the end of this thesis.

1.5 Thesis Outline

I will explain to you the details of the proposed system in the following chapters. In Chapter 2, I would like to have an overview of the existing IRIS system. Then, in Chapter 3 and 4 I will introduce the details of some mechanisms such as auto-charging, auto-scoring and auto-judging. I will show the experimental result of the result system in Chapter 5. Finally the conclusion is given in Chapter 6.

Chapter 2 The IRIS system

2.1 Hardware setup

Similar with MiroSot in FIRA, IRIS is also a vision based robot soccer system. The real time information, including ball position and robot pose, is collected by the camera located above the game field. The camera is connected to the server through IEEE 1394 (Firewire). Then, with the help of computer vision, the visual information is processed for the calculation of quantitative information such as robot position and orientation. Players can use their joysticks to send control instructions to the robots. Those instructions are converted to velocities of the wheels and then sent to the robots through the RF emitter.

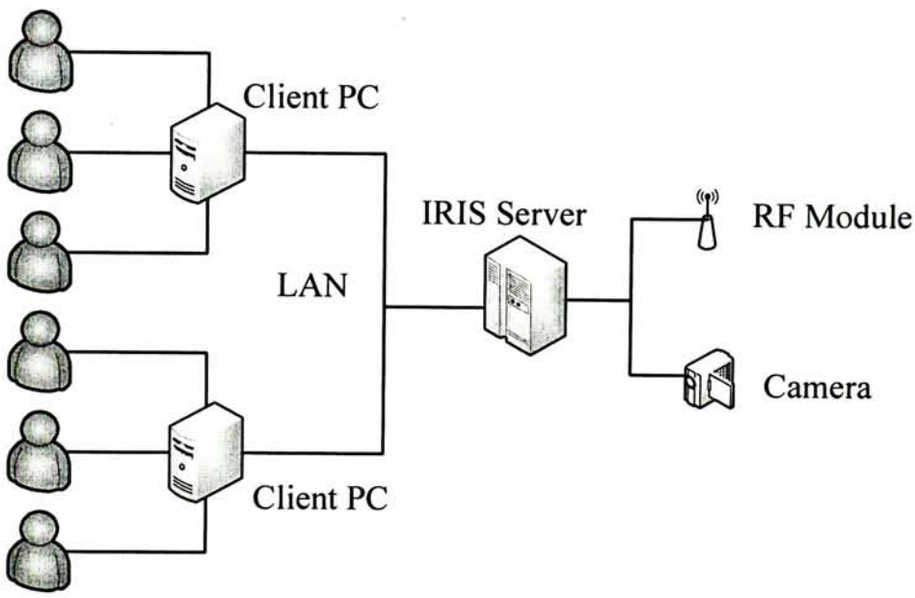


Figure 2.1 Hardware architecture of IRIS

Camera

A PTgray CMOS color camera is used that uses the ieee1394 (Firewire) interface. It produces 30 color frames per second. Image size: 640*480

Game Field

Wooden, painted green. Size: 2200mm×1800mm.

Wireless communication

IRIS selects Radio Frequency (RF) as its wireless communication module. The RF module is centered on an 8-bit MCU MC9S08GT60 produced by Freescale Semiconductor. It provides several power-down modes and an on-chip debugger. Besides, there are thousands of related resources available for this MCU which make it very suitable for our use. We use serial port to connect it with the server (Baud 19200 bps).

Server

The IRIS server is a PC under Windows XP Professional SP3 architecture, with an Inter Pentium ® 4 CPU 3.4GHz and 2GB of memory.

Client PC

The client PC is under Windows XP Professional SP3 architecture, with an Inter Pentium ® 4 CPU 2.4GHz and 1GB of memory.

Robots

There are totally 6 robots on the field. All robots are of the same size of 150mm×140mm×85mm. Those 6 robots are divided into two teams: team blue and team yellow. Each team has three players. Color patches on the top of robot indicate its team and number. The comparatively larger patch in the middle indicates the team (team patch) while the other one at the upper right corner indicates the number (member patch).

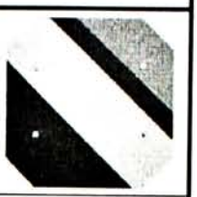
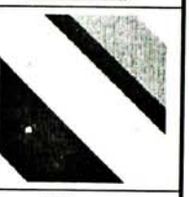
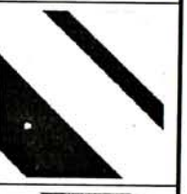
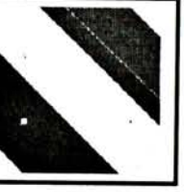
Robot	Team Patch	Member Patch	Image
Blue 1	Blue	Red	
Blue 2	Blue	Yellow	
Blue 3	Blue	Dark Blue	
Yellow 1	Yellow	Red	
Yellow 2	Yellow	Yellow	
Yellow 3	Yellow	Dark Blue	

Table 2.1 Color patches of different robots

The robots used in IRIS are differential driven. Each robot has two wheels, which are located on both sides. Each wheel is driven by a DC motor. The motion of the robots is determined by the velocities of both wheels. At any instance the robot is rotating about is called ICC (Instantaneous Center of Curvature).

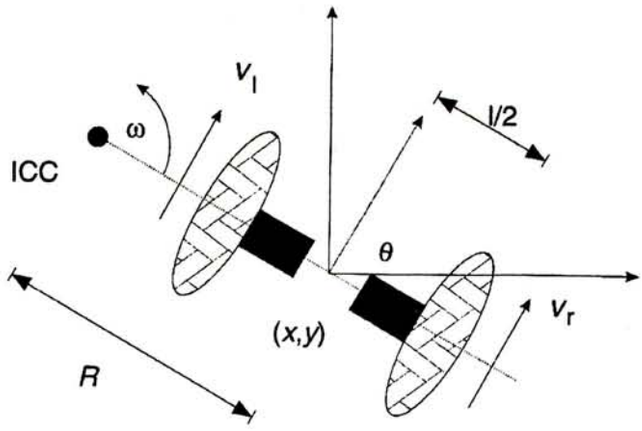


Figure 2.2 Differential Drive kinematics [12]

Because the angular velocity ω about ICC is the same for both wheels, we have [13]:

$$\begin{aligned} \omega(\mathrm{R} + l/2) &= V_r \\ \omega(\mathrm{R} - l/2) &= V_l \end{aligned} \tag{2.1}$$

where l is the distance between the centers of the two wh/ls, V_r and V_l are the right and left wheel velocities along the ground, and R is the signed distance from the ICC to the midpoint between the wheels.

We can therefore solve R and ω:

$$\begin{aligned} R &= \frac{1}{2}\frac{V_l + V_r}{V_r - V_l} \\ \omega &= \frac{V_r - V_l}{l} \end{aligned} \tag{2.2}$$

Here we have some special cases:

1. If $V_l = V_r$, then we have forward linear motion in a straight line. R becomes infinite, and there is no rotation.

2. If $V_l = V_r$, then $R = 0$, and we have rotation about the midpoint of the wheel

axis which means that it can rotate in place.

3. If $V_l = 0$, then we have rotation about the left wheel. In this case $R = 1/2$. Same is true if $V_r = 0$.

2.2 Software architecture

We also have software which helps us to link those hardware parts together. For example, we have to use computer vision to convert the vision frame of the camera to quantitative values, such as robot coordinates and orientations. An overview of the software architecture of IRIS is shown in Figure 2.4. We can see that each module has its own distinct responsibility. The communications between modules are done through interfaces predefined. It is a typical data sharing framework [14].

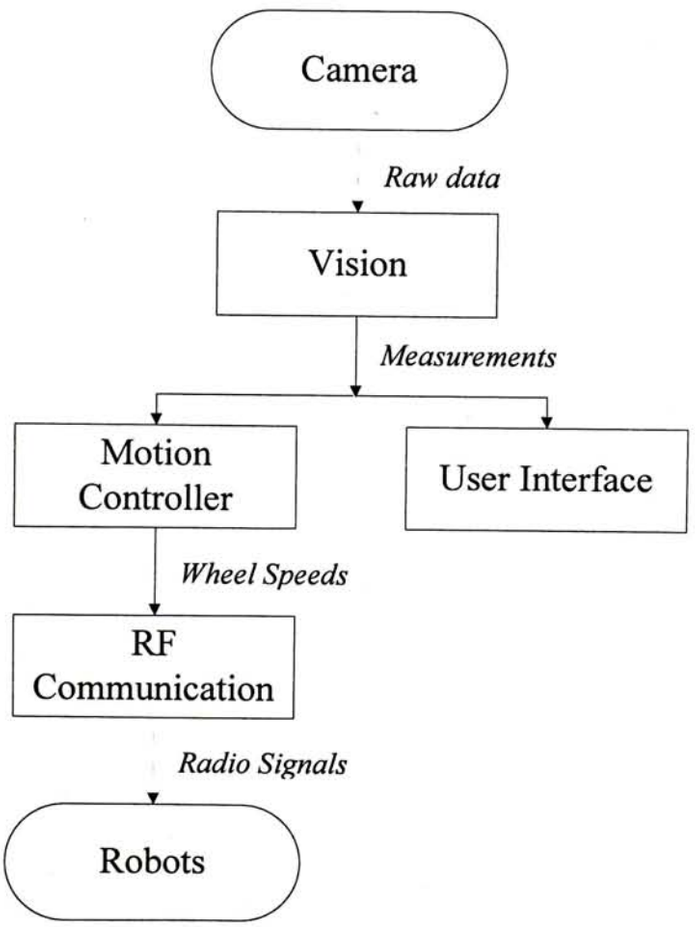


Figure 2.3 Overview of software modules of IRIS

Vision

The vision part captures the real time game field every 25ms (40fps). The raw data collected from the camera is then processed to calculate the position and orientation of the robots and ball. Firstly, background is removed. The foreground image is eroded and dilated (disk type) to remove the noise. Second, convert the image from RGB format to YCbCr format. According to ITU-R BT.601 standard:

$$\begin{aligned} Y' &= 0.257*R' + 0.504*G' + 0.098*B' + 16 \\ Cb &= -0.148*R' - 0.291*G' + 0.439*B' + 128 \\ Cr &= 0.439*R' - 0.368*G' - 0.071*B' + 128 \end{aligned} \tag{2.3}$$

Prime symbol means the gamma correction operation. Then, do the histogram statistics of colors and find the largest regions. Because the group patches has the largest coverage, we sort the regions and the top 6 largest are the group patches. Then sort the 6 patches according to the average Cb value. In our experiments, we know that Cb value of blue is larger than that of yellow, the top 3 patches with the largest Cb value are blue patches, and the rest three are yellow ones. Then, all the 6 group patches are removed from the foreground image for future use.

After this, it is time to find the members of each team. The small triangle patch beside the group patch indicates the number: red for number 1, yellow for number 2 and blue for number 3. Find the member patches of each team, and sort according to their average Cr values. The one with the highest Cr value is the red patch, which is number one. The second largest Cr value indicates the blue patch, number 3. The patch with the smallest Cr value is the yellow patch which is number 2.

I would like to mention here that we did lots of experiments to get the correct matching result just as above. However, this result may not reflect the real Cr and Cb relationships between colors. It is only correct in our case and with our color paper.

Finally we calculate the centers and orientations of the robots and the ball. The center of the robot can be calculated by the average x and y value of the total points of group patch:

$$\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$$
$$\bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \qquad (2.4)$$

where n is the total point number in the group patch. x_i and y_i are the x and y coordinates of those points. The orientation of the robot is calculated by a function *CalculateRgnAngle()* in our program. It calculates the angle according to the relative positions of group patch and member patch. Details on this topic can be found in [15].

User Interface

The traditional client interface of IRIS is as following:

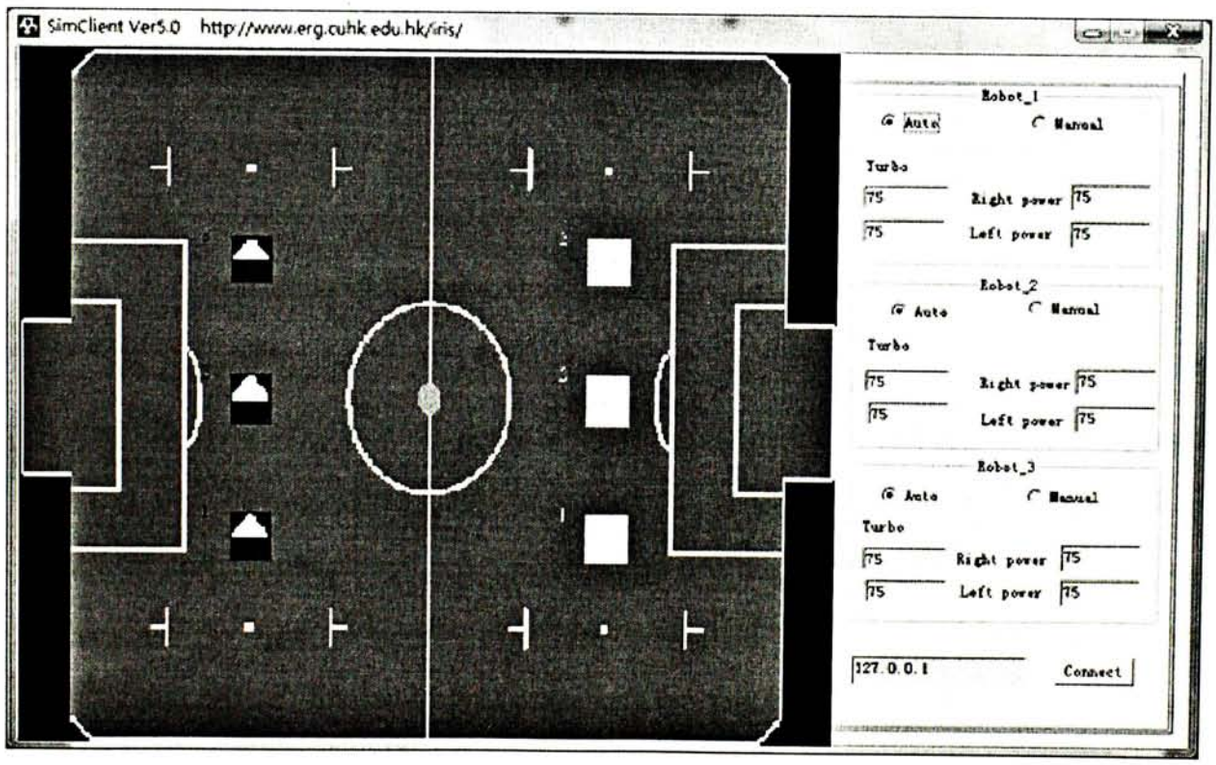


Figure 2.4 Traditional client interface of IRIS

where the left half is the real time simulated video of the game field. For each side, there are three joysticks linked to the client PC. Players can tune the power for each wheel on the right. It is connected to the server via LAN.

Because there is no game information, such as pause, start or fault, displayed in the client program, it is not easy for us to conduct the game remotely.

RF communication

The wireless RF emitter broadcast a byte array of 71 characters which includes the velocity information of 10 robots in maximum:

Array[]	Content	Array[]	Content
0	0x85	[i*4+6]	Right Wheel Velocity
1	0x8A	[i*4+7]	Button Pressed
2	0xFF	[i*4+8]	0x00
3	0x56	4,49~69	0x00
[i*4+5]	Left Wheel Velocity	70	0xFF

Table 2.2 Bytes sent by RF emitter (former IRIS version)

where i is the ID of robot ($0 \le i \le 10$). Switches are provided on each robot to define ID and RF channels. Robots will receive and analyze the data packages and read only the part indicated by its ID. The velocity value is of the Pulse Width Modulation (PWM) form with $0 \le |v| \le 100$. Positive value means that the velocity is forward while negative means backward velocity. The higher the absolute value of velocity is, the faster the wheel turns.

Figure 2.5 Switches on the robots defining robot ID and RF channels

Motion control

All robots are moved manually according to the joystick instructions. So there is not any real motion control existing. Players can define their basic velocity value (positive values required) just as Figure 2.5. Then, when different direction key are pressed, different velocity values are sent to RF module to control the robots:

Assume that the basic velocity values defined by players for both wheels are V_l and V_r:

Key pressed	Left wheel velocity	Right wheel velocity
Up	V_l	V_r
Down	$-V_l$	$-V_r$
Left	*$20-V_l$, -20 at most*	*V_r -20, 20 at least*
Right	*V_l-20, 20 at least*	*$20-V_r$, -20 at most*
Upper left	*V_l-40, 0 at least*	V_r
Upper right	V_l	*V_r-40, 0 at least*
Bottom left	*$40-V_l$, 0 at most*	$-V_r$
Bottom right	$-V_l$	*$40-V_r$, 0 at most*

Table 2.3 Velocity values sent to RF emitter

Chapter 3 Internet Accessibility

Former version of IRIS is not remote accessible. Players have to assemble in our contest venue. It is a heavy cost to hire venues and student helpers. What else, it is an obstacle on the way of the popularizing of IRIP. IRIP is an internet based innovation park. Therefore, all the components of it should be internet accessible. We have to transplant the traditional local IRIS to the internet. Then, players from all over the world can join our games by opening a website other than coming to our lab.

In the traditional IRIS, simulator is installed in every client PCs. Meanwhile, a server program is running at the server. Both of them are subject to be modified to meet the requirements of internet accessibility.

On the client end, we select Flash program to replace the former simulator. Flash is widely used on internet nowadays because of its small file size, great visual effect, good interaction with users and easy accessibility. If we put the client flash program on our website, player can access to the game by simply downloading the flash program and login. They can even play the game on their cell phones only if there terminals support Flash.

In order to communicate between server and clients, we firstly establish a socket connection. There are 6 individual flash programs to control 6 robots. Therefore, server keeps 6 socket links simultaneously during competitions. If it is time for play and no one is controlling this robot, flash will connect to the server automatically when it is opened by the player.

After the connection, we use an array of 56 bytes to communicate between server and clients.

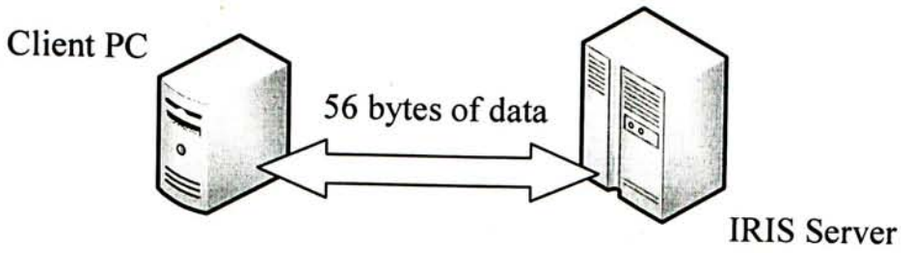


Figure 3.1 Array transmitted between server and client

When the array is sent from the server to the clients, it includes position and orientation values as well as the control information. The position and orientation values are corrected to 2 decimal places:

Array[ ]	Contents
6**i*+1	X coordinate of *i* (integer part)
6**i*+2	X coordinate of *i* (decimal part)
6**i*+3	Y coordinate of *i* (integer part)
6**i*+4	Y coordinate of *i* (decimal part)
6**i*+5	Orientation radius of *i* (integer part)
6**i*+6	Orientation radius of *i* (decimal part)

Table 3.1 Position and orientation values sent by server

where variable *i* is defined as:

i	Corresponding object	*i*	Corresponding object
0	Ball	4	Judge Robot
1	Robot Blue 1	5	Robot Yellow 1
2	Robot Blue 2	6	Robot Yellow 2
3	Robot Blue 3	7	Robot Yellow 3

Table 3.2 Relationship between variable *i* and the corresponding objects

Client will receive this array every 50ms. Flash program will renew the display according to the coordinates received.

Beside those position and orientation information, there are also some control messages to transmit. Array [0] is a special element. It helps to transmit status messages:

Value of Array[0]	Meanings
0	Game pause
13	Penalty Kick Blue
14	Free Kick Blue
15	Don't Move Robot Blue
16	Penalty Kick Yellow
17	Free Kick Yellow
18	Don't Move Robot Yellow
21	Change field (left: blue; right: yellow)
25	Change field (left: yellow; right: blue)
default	Start

Table 3.3 Status messages indicated by Array [0]

Besides Array [0], there are also some other bytes which help to display scores and time:

Array[]	Contents
50	Score of team blue
51	Score of team yellow
52	Eclipsed time (second)
53	Eclipsed time (minute)

Table 3.4 Elements in Array denoting scores and time

The array sent by the server is broadcasted to every client.

When clients attempt to send velocity information to the server, the contents of this 55bytes array change:

Array[ ]	Contents
0	Constant 100
1	Number of Robot *
2	1: if Ready button is pressed 2: if Pause button is pressed
4	Right wheel PWM value
5	Left wheel PWM value

Table 3.5 Array sent by clients to the server

* The numbers of robots are defined as:

Number	Robot	Number	Robot
0	Blue 1	5	Yellow 1
1	Blue 2	6	Yellow 2
2	Blue 3	7	Yellow 3

Table 3.7 Numbers of robots

There are 6 individual flash programs to control 6 robots. Therefore, server keeps 6 socket links simultaneously during competitions. If it is time for play and no one else is controlling this robot, flash will connect to the server automatically when it is opened by the player. Socket connection is linked to port 23 of the server. As it is a port used usually for telnet connection, it is not always blocked by firewalls. We also have to pay attention to the security policy of flash to avoid connection failure. Here,

we must set the IP address of our IRIS server (137.189.100.85 or 137.189.100.124) to trusted addresses. We also have the following sentence in flash program:

Security.loadPolicyFile("xmlsocket://137.189.100.124:21");

It loads a cross-domain policy file from the IRIS server. Flash Player use this policy file to determine whether to permit applications to load data from servers.

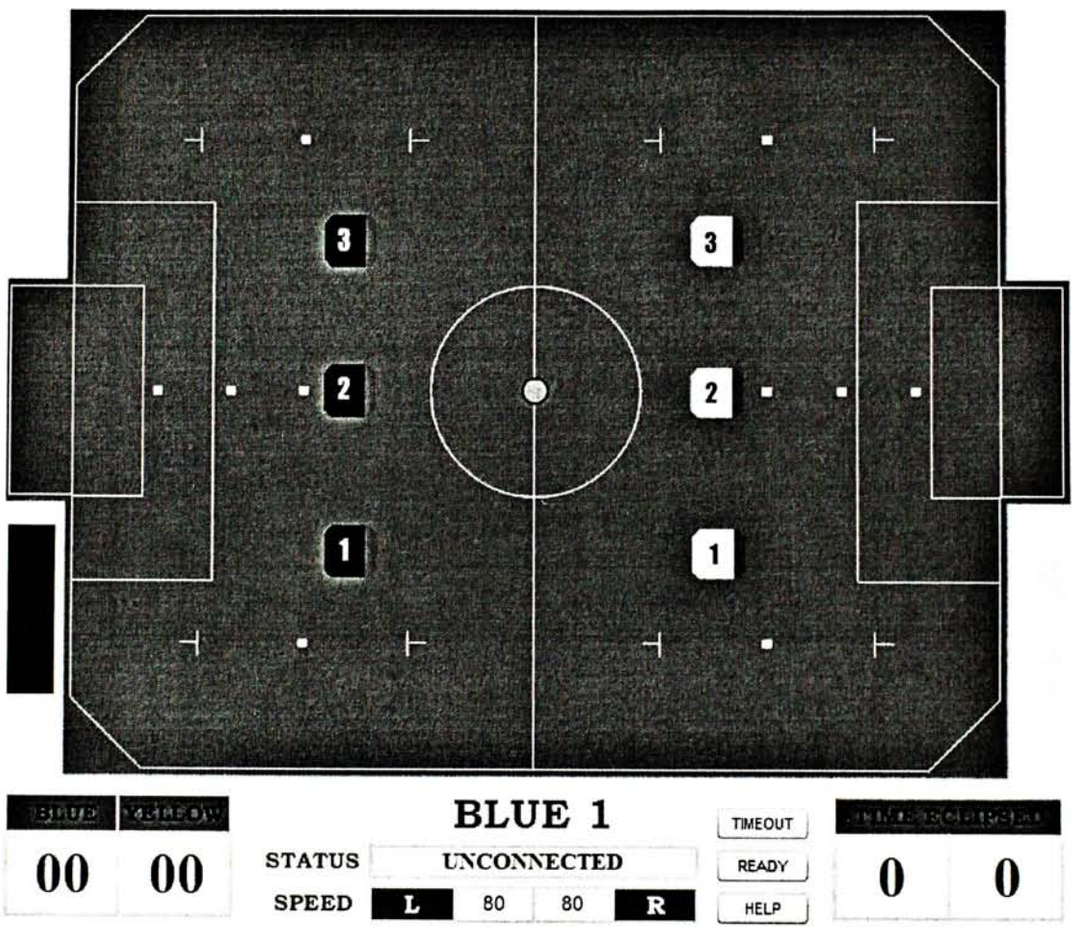


Figure 3.2 The flash program for IRIS

After the flash status window display 'CONNECTED', players can press READY button. When all six clients are ready, judge starts the contest. Then, the content of the status window changes to 'Start'. Players can also require timeout when contest is going by pressing TIMEOUT button. They can tell the judge the reason for

timeout by external voice software or the integrated chatting window. Players can also see each others as well as the real time video of the game field.

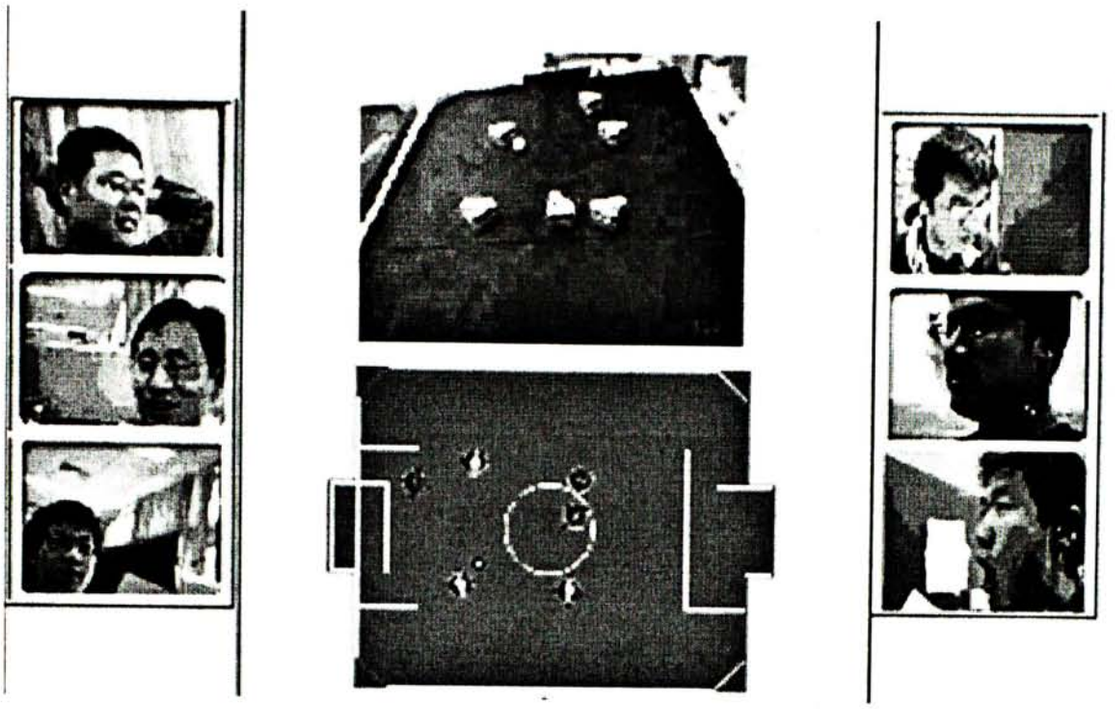


Figure 3.3 User interface with videos of players and game field

Chapter 4 Auto-charging

Now it is time to liberate us from the heavy and boring maintenance work. One of the most important issues is the auto-charging mechanism. Previously, changing the batteries for robot cost a lot of time. We have had some statistics that to change batteries for one single robot would cost about 45s to 1min for a skilled staff. In order to keep the fairness and efficiency of the competitions, we have to change the batteries for all 6 robots after every 10 minutes competition, which is really a boring task.

In the expected improved version of IRIS, such function is to be done automatically. We have to think about what modifications should be done on the existing IRIS system.

Begin with hardware. Previously, the robots only received the broadcasted 71 bytes from the RF emitter (Table 2.2). Such communication is one way, from the RF emitter to robots. Now, in order to know whether a robot is low-power, we need to enable the robots to report their voltage level when request. Because the voltage is an analog signal, we firstly convert it to a digital signal with voltage A/D converter. Also, the robots need RF module to report their voltage and the formal emitter needs to be switched to full duplex work mode in order to send while receiving. Of course, the most important thing is to build a new game field with charging chambers as well as charging mechanisms.

Software also needs to be improved. For the vision part, previous IRIS always assumes that 6 robots and a ball are on the field. If the amount of robot or ball is no correct, errors will occur. But in the proposed IRIS, there will surely be violated cases when changing the robots: a fully charged robot goes out of the charging chamber while its counterpart, the low-power one, has not yet get into the chamber. The vision module is to be modified therefore. Another module is motion control. Previous IRIS has no ability of motion control because control is done manually by the players. However, in the new system, some motions cannot be controlled manually such as

driving the low-power robot into the charging chamber etc. There must be an automatic mechanism to drive the robots to desired positions. The mechanism is just motion control.

Two other important issues are the auto-judging and auto-scoring mechanisms. Details of them are introduced in the next chapter.

We can see an overall flowchart of the automated IRIS system as Figure 4.1.

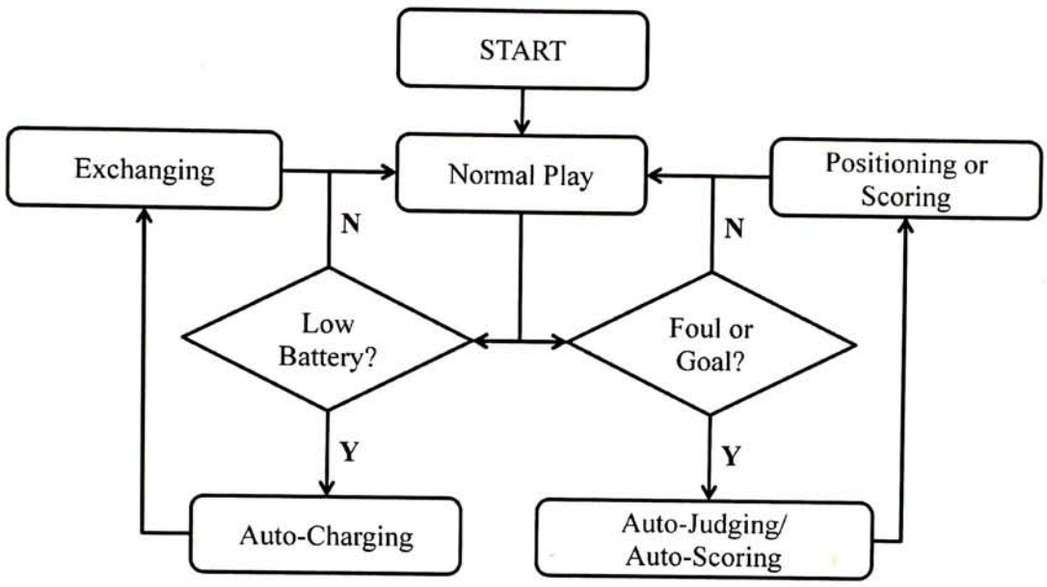


Figure 4.1 Overall flowchart of the automated IRIS system

4.1 Hardware setup

For the new system, our IRIP group built a new game field:

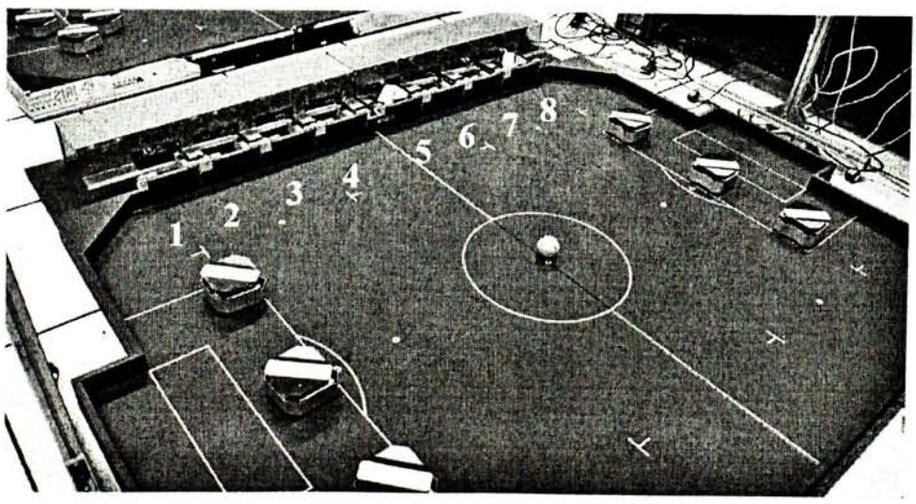


Figure 4.2 New game field built for Automated IRIS and chamber numbers

It is similar to the previous one (see Figure 1.2). The only difference is the 8 charging chambers on one side (see Figure 4.2). There are 7 charging plugs for chamber 1to 7. Chamber 8 is for standby without power supply. For every chamber there is a lock to lock the door, a joint that opens and closes the door and a switch to indicate that the robot has successfully entered the chamber. All those mechanicals are controlled by an on-chip computer. Such system we call it Auto-Charging System (ACS).

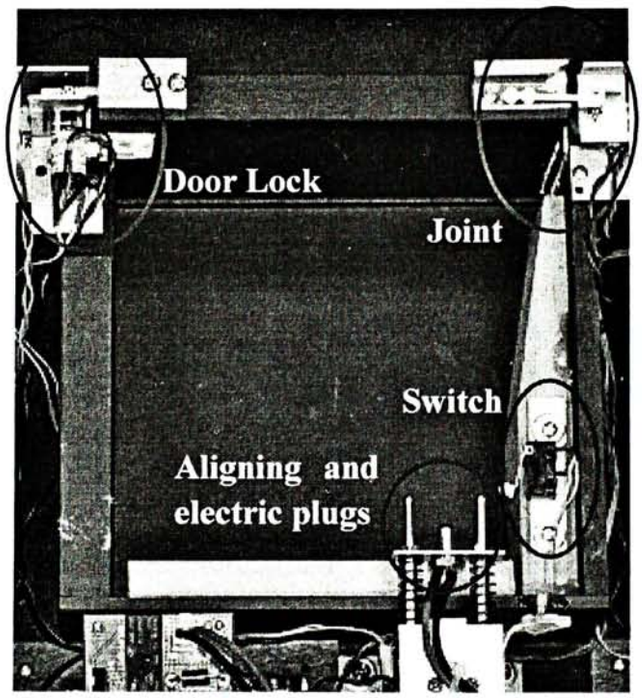


Figure 4.3 The charging chamber

Both the joint and lock are driven by DC motors. For the locker, a shaft is linked to the DC motor. It can rotate with the motor. There is a slot on one side of the door. When there is an instruction to close the door, the motor starts to rotate. Once the shaft entered the slot, the door is closed. To open the door, drive the motor inversely. The switch can be closed by the entering robot. When it is closed, the system knows that such robot has successfully entered the chamber. Door is then closed and the charging is about to start.

Meanwhile, the previous robots are also to be redesigned. Power level monitor

module is added on each robot. It can reply the realtime power level to the server when required. There are 3 sockets on the back of each robot (see Figure 4.4). The black one in the middle is the electric socket. The other two beside are for alignment. In order to facilitate charging, when the robots enters a charging chamber, the aligning plugs of the auto-charging system (see Figure 4.3) is inserted to the aligning sockets, which can make it very easy for the inserting of the electric plug (Figure 4.5).

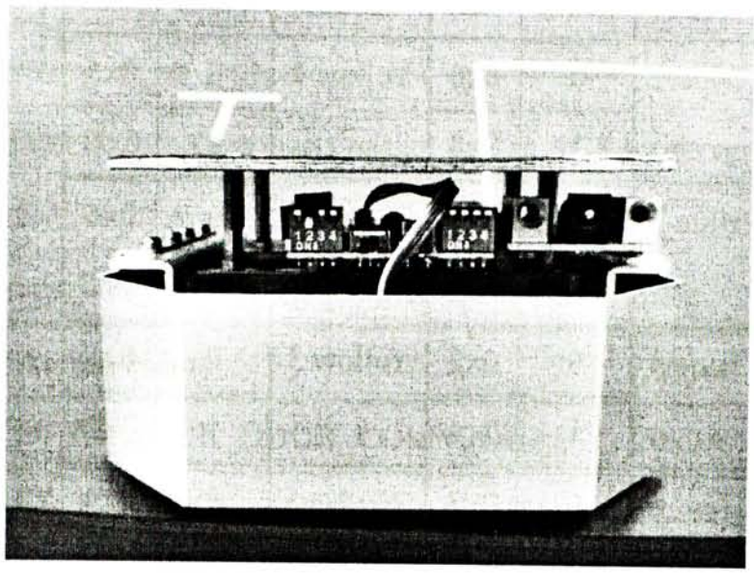


Figure 4.4 The robot for new system

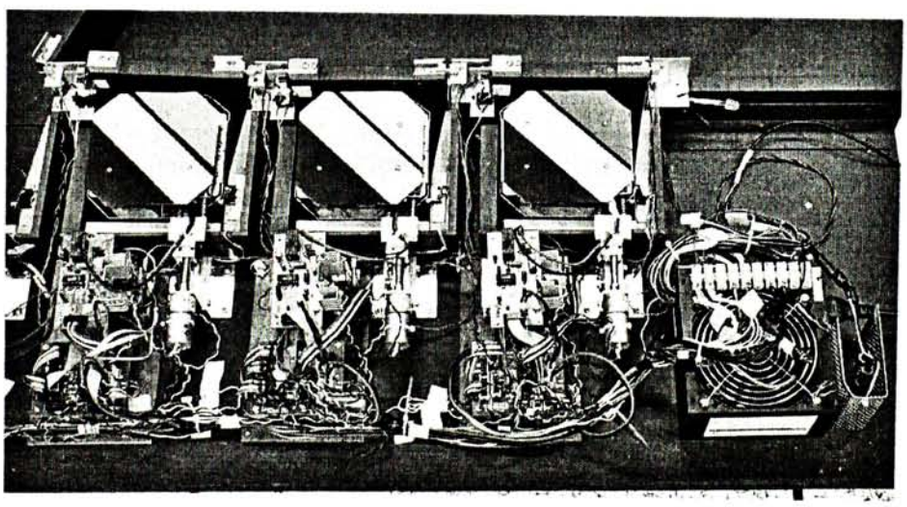


Figure 4.5 Robots inside the charging chambers

We also added another set of robots for exchanging. Now we have two sets of robots,

totally 12 of them. For auto-judging mechanism, we also added two judge robots. During the game, if one robot is on the field, its counterpart in the other team has to standby in the charging chamber. Accordingly, we reassigned the IDs to each robot:

Robot	Team	ID	Robot	Team	ID
Blue 1	0	0x0	Yellow 1	0	0x5
	1	0x8		1	0xB
Blue 2	0	0x1	Yellow 2	0	0x6
	1	0x9		1	0xC
Blue 3	0	0x2	Yellow 3	0	0x7
	1	0xA		1	0xD
Judge	0	0x3	Judge	1	0x4

Table 4.1 ID of Robots

4.2 Communication

In the new IRIS system, there are 2 series connections to be established between server and ACS/RF emitter. The link between server and ACS is newly added. It is used to control the auto-charging. ACS is connected to the server by a USB2COM dongle (Baud 19200). The protocol of communication between them has been defined as followings. First, we defined the bytes sent from the server to ACS:

Byte	Content	Meaning
1	0x85	Start Prefix 1
2	0xAA	Start Prefix 2
3	0x01~0x08	Active Object: Chamber 1 to 8
4	Action Command*	Indicates what action is to be done
5	0x55	End Suffix 1
6	0xFF	End Suffix 2

Table 4.2 Command packet sent to ACS by server

* Action Command:

Action Name	Value
OPEN_LOCK_DOOR_COMMAND	0x0A
CLOSE_LOCK_DOOR_COMMAND	0x03
LOCK_OPEN_COMMAND	0x1A
LOCK_CLOSE_COMMAND	0x13
DOOR_OPEN_COMMAND	0x2A
DOOR_CLOSE_COMMAND	0x23
CHARGER_HEAD_IN_COMMAND	0x3A
CHARGER_HEAD_OUT_COMMAND	0x33
CHARGER_START_COMMAND	0x4A
CHARGER_STOP_COMMAND	0x43
READ_VOLTAGE_COMMAND	0x6A
AUTO_CHANGE_ROBOT_COMMAND	0xCC

Table 4.3 Action command list (server to ACS)

While, the response of ACS has 25 bytes as:

Byte	Content	Meaning
1	0x85	Start Prefix 1
2	0xAA	Start Prefix 2
3	0x01~0x08	Active Object: Chamber 1 to 8
4	System Status*	Indicates the real time status of ACS
5-12	0x00 or 0x01	Detail status of ACS (status of each component)
13~18	Voltage Values	Battery Information
19	Error Report#	Indicates whether the previous action is success or not
20~23	Null	Reserved
24	0x55	End Suffix 1
25	0xFF	End Suffix 2

Table 4.3 Response packet (ACS to server)

* System Status:

Action Name	Value	Action Name	Value
REPORT_STATUS	0x77	ROBOT_IN_AGAIN	0x13
FINISH_OPERATION	0x99	AUTO_CHARGE	0x14
IDLE_STATE	0x00	CHARGE_FAST	0x15
AUTO-STATE	0x10	CANNOT_FAST_CHARGE	0x16
ROBOT_STILL_IN	0x11	CHARGE_TRICKLE	0x17
ROBOT_WAS_OUT	0x12		

Table 4.4 System status list

Error Report:

Meaning	Value
SUCCESS	0x77
FAILED	Other Values

Table 4.5 Error report list

Here, for example, if we want chamber 1 goes to "auto change robot command", the packet will be:

0x85 0xAA 0x01 0xCC 0x55 0xFF

Then chamber 1 will stop charging, charging head will move to standby, and then the door will open. Return message is 25 bytes long, and will be sent back from ACS to Server after the door opens:

0x85 0xAA 0x01^ 0x10 0x01 0x01 0x0 0x01 0x0 0x0 0xA5 0xA5 0xA5 0xA5 0xA5 0xA5*
0xA5 0x77# 0x0 0x0 0x0 0x0 0x55 0xFF

where:

0x01^ : Charger number is 1

0x10* : System status is AUTO_STATE

0x77# : Error report is SUCCESS

Then, server drives the robot that needs charging into chamber 1 as our example. Chamber door will be closed and locked automatically when the robot hit the switch, then charging action will start. Return message from chamber 1 will be sent as shown below:

0x85 0xAA 0x01 0x15 0x02 0x02 0x0 0x0 0x0 0x01 0x01 0x01 0x0 0xEA 0x01 0x4F*
0x02 0xBD 0x77# 0x0 0x0 0x0 0x0 0x55 0xFF

where:

0x15* : System status is CHARGE_FAST

0x77# : Error report is SUCCESS

The very beginning of the charging is fast charging. The chips inside the ACS system will monitor the voltage of the batteries in all time and switch to trickle charging mode when the voltage reaches a certain level. At this instance, ACS will also response a news with the system status equals 0x17 which means that currently the charging mode is trickle charging.

Another connection is the link between server and RF emitter. Because we have added another set of robots, the RF emitter is reprogrammed to send 74 bytes each time. It also receives the voltage response from the robots.

Array[]	Content
0	0x85
1	0xAA
[*i**4+2]	Left Wheel Velocity
[*i**4+3]	Right Wheel Velocity
[*i**4+4]	Button Pressed
[*i**4+5]	0x00
[*i**4+6]	Report Control
72	0x55
73	0xFF

Table 4.6 Command packet from Server to RF emitter (after extension)

where i is the ID of the robots ($0 \leq i \leq 14$). Server set the corresponding report control byte to REPORT_VOLT <0xA5> and set byte 72 to 0xAA for reading the voltage of a robot in game field. USB RF emitter will switch between Tx and Rx state automatically in order to receive message while sending.

Server will receive data from robot via USB RF. Data is one byte long. The relationship between data received and the real voltage is:

$$voltage=(data\_received*1.2-2)/31 \quad (4.1)$$

We can determine whether to change it or not according to the voltage value we received. However, for convenience sake, all robots are to be changed after every competition.

4.3 Vision

We have discussed that the vision part of the previous IRIS can only recognize 6 robots on the field. Otherwise it may not output the right result. However, in the proposed new system, when changing robots, there will surely be more than 6 robots on the game field at some times. So we need to improve the vision part in order to meet our requirement.

Here, we have a problem: take robot Blue 1 for example (see Figure 4.5). Robot Blue 1 in team 0 has absolutely the same color patches at the top with its counterpart in team 1. Imagine that Robot Blue 1 of team 1is on the game field. Its counterpart (team 0) is charging in chamber 1. Later, the battery level of robot Blue 1 (team 1) is low. ACS has decided to drive it into the charging slot for charging. Then, ACS opens the door of the chamber 1 and drives the other robot Blue 1 (team 0) out of it. At this moment, there are two absolutely same robots on the game field. Then, how to distinguish them is very important as the server has to know which one it is controlling.

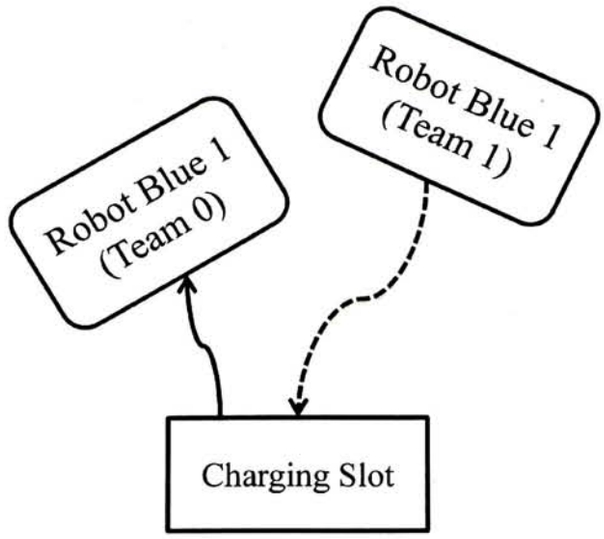


Figure 4.6 Co-occurrences puzzle

We found a good way to go which avoided confusion: When changing the robot, firstly, remember the IDs of robots on the game field and record their team numbers. In our system, we store them in a Boolean array: status[i] ($0 \leqslant i \leqslant 6$):

Status[i]	Corresponding Robots
0,1,2	Blue 1, 2, 3
3	Judge
4,5,6	Yellow 1,2,3

Table 4.7 Corresponding robots of array Status

where the value status[i] denotes which group is on the game field or which group is to be controlled currently. For example, status[i] = 0 means that group 0 of robot i is being controlling by ACS. Then, mask the robots which are to be changed by setting them to the background as the background information will not be processed in the following steps. We call a function *Cgrab::ForeBackGroundLabel(float fThres)* to do this. Firstly, we store the centers (x,y) and orientations θ of the robots which are to be charged. Then, we can calculate the vertexes of the robot coverage rectangle areas as following:

$$
\begin{aligned}
&point1.x=x+side*cos(\theta)-side*sin(\theta);\\
&point1.y=y+side*sin(\theta)+side*cos(\theta);\\
&point2.x=x+side*cos(\theta)+side*sin(\theta);\\
&point2.y=y+side*sin(\theta)-side*cos(\theta);\\
&point3.x=x+-side*cos(\theta)+side*sin(\theta);\\
&point3.y=y+-side*sin(\theta)-side*cos(\theta);\\
&point4.x=x+-side*cos(\theta)-side*sin(\theta);\\
&point4.y=y+-side*sin(\theta)+side*cos(\theta);
\end{aligned}
\tag{4.2}
$$

where *side* is the side length of robot. In our system, the value is 18 pixels. Later, we can draw several polygons indicating the coverage areas of those robots. Then, we can do judgements for every point in the vision frame to see whether it locates inside those polygons. If so, we can set it to the background to ignore it. So, after this, only the robots which are no need to be changed are left in the vision frame.

Lastly, drive the fully charged robots out of the chamber, the system will automatically find them. Wait until they stop, mask these fully charged robots instead, and drive the low-power ones into the chambers.

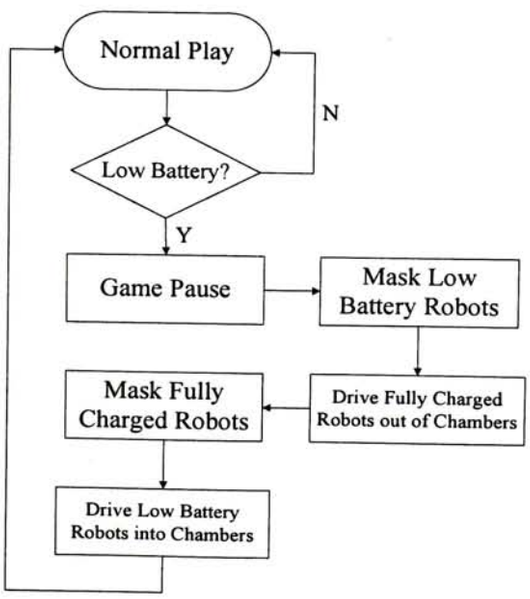


Figure 4.7 Flowchart of vision module in Auto-Charging

4.4 Motion control

Motion control is a key role in the entire system. Its performance will largely affect the efficiency of the whole system. Because it will be used in many cases, such as driving the robots into the chamber, judge robot fetching the ball, repositioning of robots when fault, etc.

The goal of motion control is to drive the robot to a desired position and orientation without hitting any obstacles and do this in an efficient way [16]. There are two kinds of motion control methods: deliberative motion and reactive motion [1]. Deliberative motion usually involves the generation of an explicit plan that describes how to reach a target location. It may be more consistent and able to realize long term goals, while it is not very suitable for the highly dynamic robot soccer case because it would fail on the planned path and spend a lot of time on re-planning. While reactive motion contains no planning in advance. It calculates in real time according to the states and the inputs. Here, we choose reactive motion since it fits our case better. It is quick and can even avoid collision because of the online calculation. Although the final path may not be optimistic, it is very useful in such a high dynamic environment.

Here, in IRIS, we would like to have two kinds of motion controller. One is Artificial Potential Field (APF); the other is the Close-loop Reactive Control (CRC). Because of the features of robot soccer, collision avoidance is more important than short path. Those two motion controllers above are used in different situations. APF is suitable for cases when there are obstacles on the expected trajectories of robots. CRC is used when obstacle avoidance needs not to be considered. APF is extensively used in robot trajectory planning [17]. However, the accuracy and efficiency is comparatively low. So in cases requiring high accuracy we turn to use CRC instead.

4.4.1 APF

APF is sourced from the physical potential fields. It has been widely used in mobile

robot motion control because of its elegance and intuitive simplicity [18, 19, 20, 21, 22]. Khatib defined that the potential field has a minimum at the goal while potential hills at all obstacles [19]. Robot in such a potential field will be attracted by the goal while repelled by the obstacles. Goodrich gave a very good tutorial on this [23].

There are many types of potential field function. One kind is called the local potential approach which needs only the local information to calculate the potential field. Such APF is generated by combining the attractive potential fields and repulsive potential fields together [12, 13, 24, 25, 26, 27]. The main drawback of this local approach is the local minima: the robot would be 'trapped' at a point away from the goal. Another disadvantage is that it is hard to predict the trajectory [28]. Because of those drawbacks existing, researchers developed more approaches to get rid of these drawbacks. The Harmonic Potential Functions [20] attempts to get a result potential field without the problem of local minima. More factors such as velocities of robots, obstacles and targets are also considered in [18]. This will improve the performance in the experiments.

At the very beginning, we choose to use the basic approach: the one introduced by Goodrich [23], because it is simple enough and have good performance in experiments. Just as what has been mentioned above, APF is generated by combine the attractive potential fields and repulsive potential fields together. The procedures are as follows:

Attractive potential field

Let (x_G, y_G) denote the goal coordinate and r denote the radius of the goal. (x,y) is the coordinate of the robot. According to Figure 4.7, we find the distance d between the goal and the robot:

$$d = \sqrt{(x_G - x)^2 + (y_G - y)^2} \tag{4.3}$$

And the angle between the robot and the goal:

$$\theta = a\tan 2((y_G - y),(x_G - x)) \tag{4.4}$$

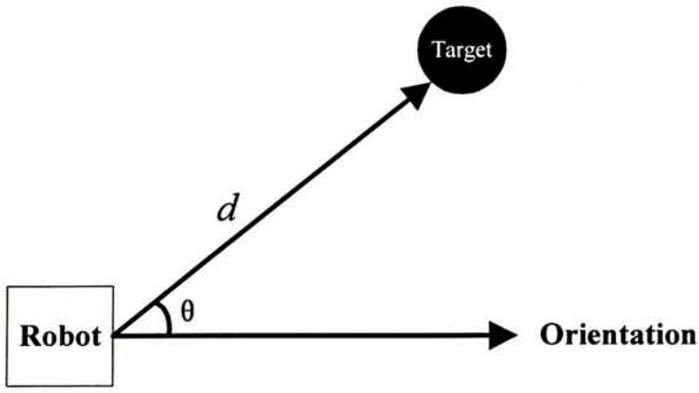


Figure 4.8 Position and orientation of robot w.r.t the target

Then the forces alone x and y axis can be determined by:

	Δx_{att}	Δy_{att}
$d<r$	0	0
$r<d<r+s$	$\alpha(d-r)\cos\theta$	$\alpha(d-r)\sin\theta$
$d>r+s$	$\alpha s\cos\theta$	$\alpha s\sin\theta$

Table 4.8 Attractive forces along x and y axis

Here, the goal is treated as a circle with radius *r*. *s* is the spread of the attractive field and α is the parameter for tuning the strength of the field ($\alpha > 0$). At the points inside the goal ($d<r$), the attractive potential field is zero. But for the points outside the extend of the field ($d>r+s$), the strength of attractive potential field is to the maximum which is tuned by α.

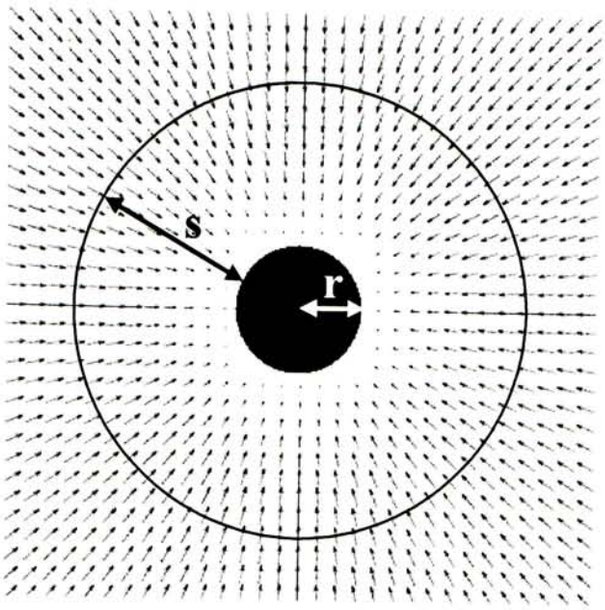


Figure 4.9 Attractive potential field

Repulsive potential field

The procedure is similar. First we calculate θ and d:

$$\theta = a\tan 2((y_o - y),(x_o - x))$$
$$d = \sqrt{(x_o - x)^2 + (y_o - y)^2} \tag{4.5}$$

where (x_o, y_o) denotes the coordinates of obstacles.

Repulsive forces are to be calculated as:

	Δx_{rep}	Δy_{rep}
$d<r$	$-sign(\cos\theta) * MAX$	$-sign(\sin\theta) * MAX$
$r<d<r+s$	$-\beta(s+r-d)\cos\theta$	$-\beta(s+r-d)\sin\theta$
$d>r+s$	0	0

Table 4.9 Repulsive forces along x and y axis

where r is the radius of obstacles. The strength of repulsive potential field is to the maximum (or even infinite) within the area of obstacle ($d<r$). If the points locate out

of the extend of repulsive force, field strength is equal to zero ($d>r+s$). The constant β is for tuning the strength of the repulsive field.

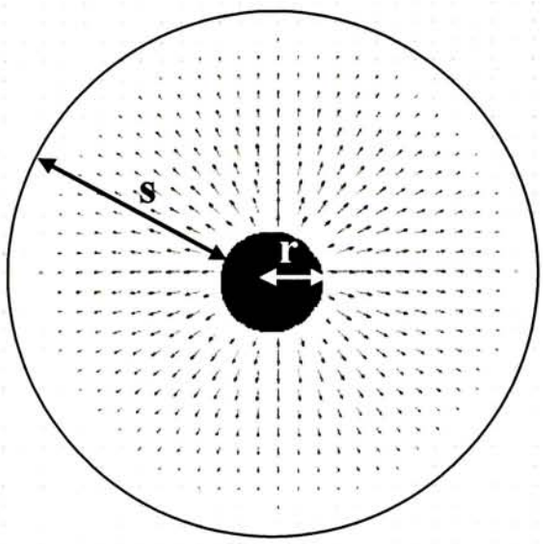


Figure 4.10 Repulsive potential field

Combination

After we have generated the attractive potential field as well as the repulsive one, the next issue is to combine the two potential fields together. The result potential field is the vector sum of all attractive and repulsive potential fields:

$$\begin{aligned} \Delta x &= \sum \Delta x_{att} + \sum \Delta x_{rep} \\ \Delta y &= \sum \Delta y_{att} + \sum \Delta y_{rep} \end{aligned} \tag{4.6}$$

Figure 4.11 Combination of artificial potential fields

As the result potential field above, the robot will be influenced in the potential field and controlled by the field. We can define that the velocity of the robot is in direct proportion to the field strength it locates. Thus we have:

$$v = \sqrt{\Delta x^2 + \Delta y^2}$$
$$\theta = a\tan(\Delta y / \Delta x) \qquad (4.7)$$

where v is the magnitude of velocity while θ is the direction of velocity. The velocity is converted into PWM values which are then sent to the robot by RF emitter. In our design, the robots always compare their own orientation with the velocity directions. If the absolute difference is larger than a threshold ($\pi/10$ for instance), the robot stops and turn to the desired orientation alone the direction of potential field. Otherwise, the robot keeps going forward.

This attempt is simple and adapt to our system because the motion as well as turning is rapid enough. We can also convert the velocity directly to the wheel speed without stopping and turning. This is to be realized in the next edition of IRIS.

4.4.2 CRC

CRC stands for Close-loop Reactive Control which is developed by Manuela Veloso et al of Carnegie Mellon University [29]. This controller helped them to won the champion of Robocup-98. It is very simple and performs well when no obstacle is located on the expected motion path.

To drive the robot into the charging chambers is not easy because the chamber is very narrow and there is no feedback from the motors of robots. In our case, there is only a gap about 2 cm on each side when a robot is entering the chamber (see Figure 4.12). All what we have is the visual feedback. However CRC is robust enough to solve those problems to conduct a quick, accurate and reliable motion control. It also has a quicker convergence than APF approach.

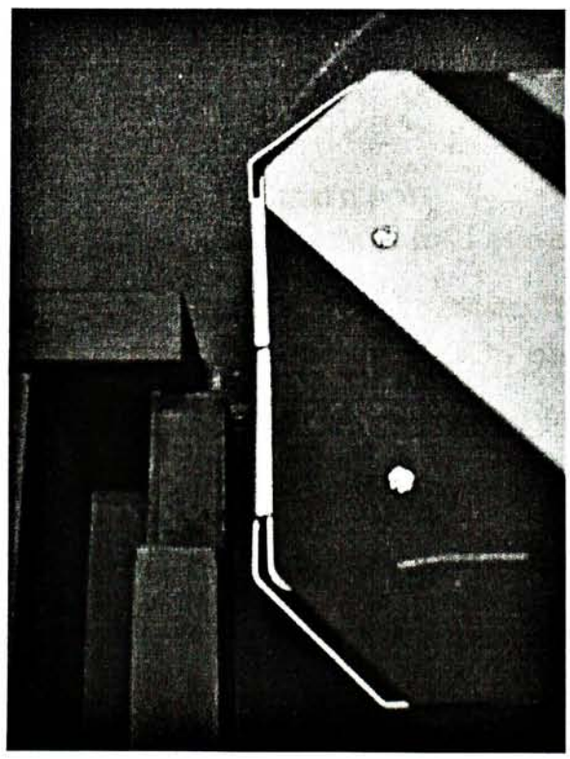


Figure 4.12 Narrow gaps when robot entering the chamber

Angle θ denotes the direction of target with respect to the robot orientation. We have t and r where:

$$(t,r) = (\cos^2\theta \cdot \mathrm{sgn}(\cos\theta), \sin^2\theta \cdot \mathrm{sgn}(\sin\theta)) \tag{4.8}$$

We can have the velocities of two wheels as:

$$\begin{aligned} v_l &= v(t-r) \\ v_r &= v(t+r) \end{aligned} \tag{4.9}$$

where v ($v>0$) is the desired speed (PWM rate in our case) set by users. The velocity is only related to the angle θ, so an external logic is to be added to this controller to judge whether the robot reaches the goal or not.

The absolute value of speed of both wheels will not exceed v since $|t \pm r|$ has been restricted to [0, 1]. It is very quick since at least one wheel will achieve the maximum speed:

- when t and r have different signs, $v_l = v(t-r) = \pm v$
- when t and r have the same sign, $v_r = v(t+r) = \pm v$

Another feature is that it can have both forward and backward velocity which is much more fast and flexible.

4.5 Processing Schemes

Now we have had all hardware and software to do auto-charging. It is time to arrange them to work together. In the first edition of ACS, we provide two kinds of charging mode: charge one robot or charge all six robots.

First of all, players on the field and the chamber coordinate must be set. Check which team the robots belong to and set them in the 'Players on the field' area (see Figure 4.12). This is very important since we have two groups of robots. The server has to know which one it is controlling especially in the cases when two robots are both on the field (during changing). And the coordinates of the chambers is also

important. Because of the design of the IRIS system, camera is not fixed. So there are always some fluctuations. We have to reset the coordinates of the center point in front of each chamber because even a small error would also influence the result at last. We have estimation that one pixel in vision equals to 4.1mm on real field. Such error would surely resist the robot from getting into the narrow chamber. So at the beginning of each game, we need to reset all chamber locations. Select the goal chamber and put the robot Blue 1 right in front of the chamber 1, then the coordinate will be displayed and press 'SET' to memorize it.

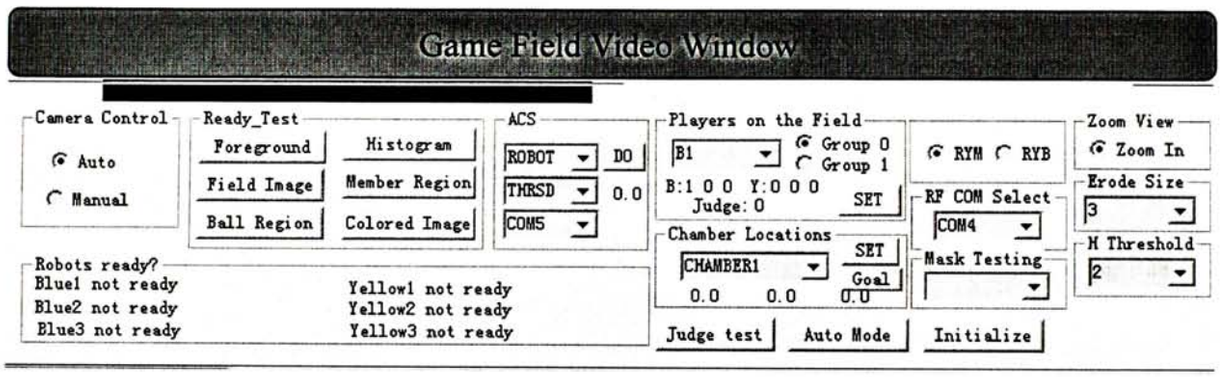


Figure 4.13 Functional modules of IRIS server

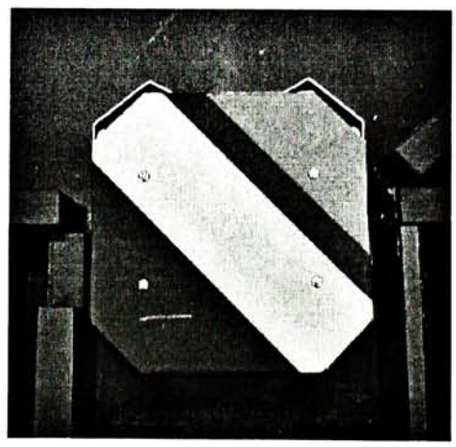


Figure 4.14 Robot Blue 1 right in front of the chamber 1

In order to avoid confusion, we have fixed the charging chamber for each robot:

Chamber #	Robot	Chamber #	Robot
1	Blue 1	5	Yellow 1
2	Blue 2	6	Yellow 2
3	Blue 3	7	Yellow 3
4	Judge (team 0)	8	Judge (team 1)

Table 4.10 Chamber number for each robot

After the preparations above we can begin to do the auto-charging. The ACS area of the server interface provides some combo boxes and button for the user to charge one robot or just check the voltage of the robot. Firstly you select which one you would like to check or change. Then, choose the threshold voltage, if the responded voltage value is lower than the threshold, change the robot. 'Always' means that always do the changing no matter what the voltage is. While 'DISP' means that only voltage value will be displayed, no changing activity is processed. The third combo box is the COM port number connected to the ACS, and the default value is COM5. After this, 'DO' button is pressed to perform the auto-charging for one robot.

The proposed exchanging procedures are as follows:

1. Stop the motion of all robots;
2. Mask the robot (low-power) in the video input;
3. Send commands to open the door of the chamber, drive the robot (fully charged) out and then hide it in the video input;
4. Unmask the robot (low-power) and drive it into the chamber;
5. Resume the robot (fully charged) and drive to desired position.

The mask of the robot is done by setting the robot area to the background thus it will not be processed. Because the distribution of robots on the game field is random, the obstacle avoidance is to be taken into consideration. Consequently, we use APF in step 3 when driving the robot out of the chamber to a desired place (always near the upper edge).

The motion control of step 4, driving the low-power robot into the chamber, is divided into a few steps. Firstly the robot is driven from the original location (point 1) to a half way point (point 2) which has the same x coordinate with the destination while the y coordinate differ 20 cm with the goal. We use APF during in this stage because we must consider the issue of collision avoidance. After the robot reaches point 2, turn its back to the goal and then enter the chamber with the control of CRC. A distance of about 20 cm is allowed for the robot to amend the errors of location when entering the chamber. After the robot reached the preset point indicating the chamber location, adjust the pose of the robot and make it exactly back to the chamber. Then, give a comparatively higher velocity to both wheels and drive it into the chamber. The door of the chamber is closed automatically when the inserting robot hits the switch at the back of the chamber.

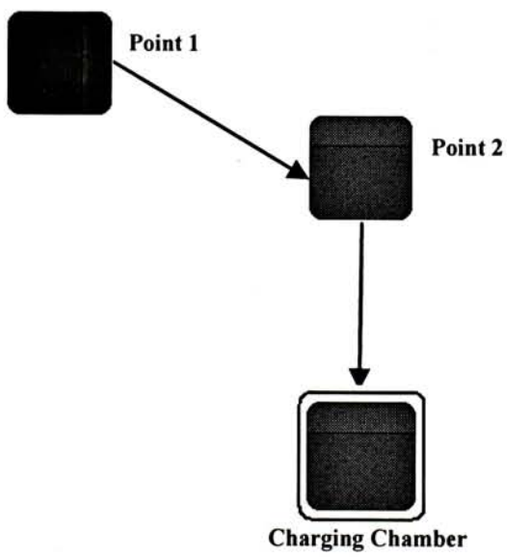


Figure 4.15 Two steps to drive a robot into the charging chamber

Another mode of auto-charging is exchanging all 6 robots. This is performed after every competition. The procedure is similar to what has been introduced in the single case. Because of the design of the game field, entrance will be blocked by the opened door of the chamber on its left. We divide the 6 robots into two groups; the ones whose chamber is not adjacent are classified into one group. Each time we exchange

one group:

Group	Members
0	Blue 1, Blue 3, Yellow 2
1	Blue 2, Yellow 1, Yellow 3

Table 4.11 Groups when changing all six robots

For higher efficiency, before changing, we can move all robots to their initial positions (Figure 4.15) in order to lower down the probability of collision. Then those fully charged ones are driven out of the chambers and guided to zone 1 by AFP.

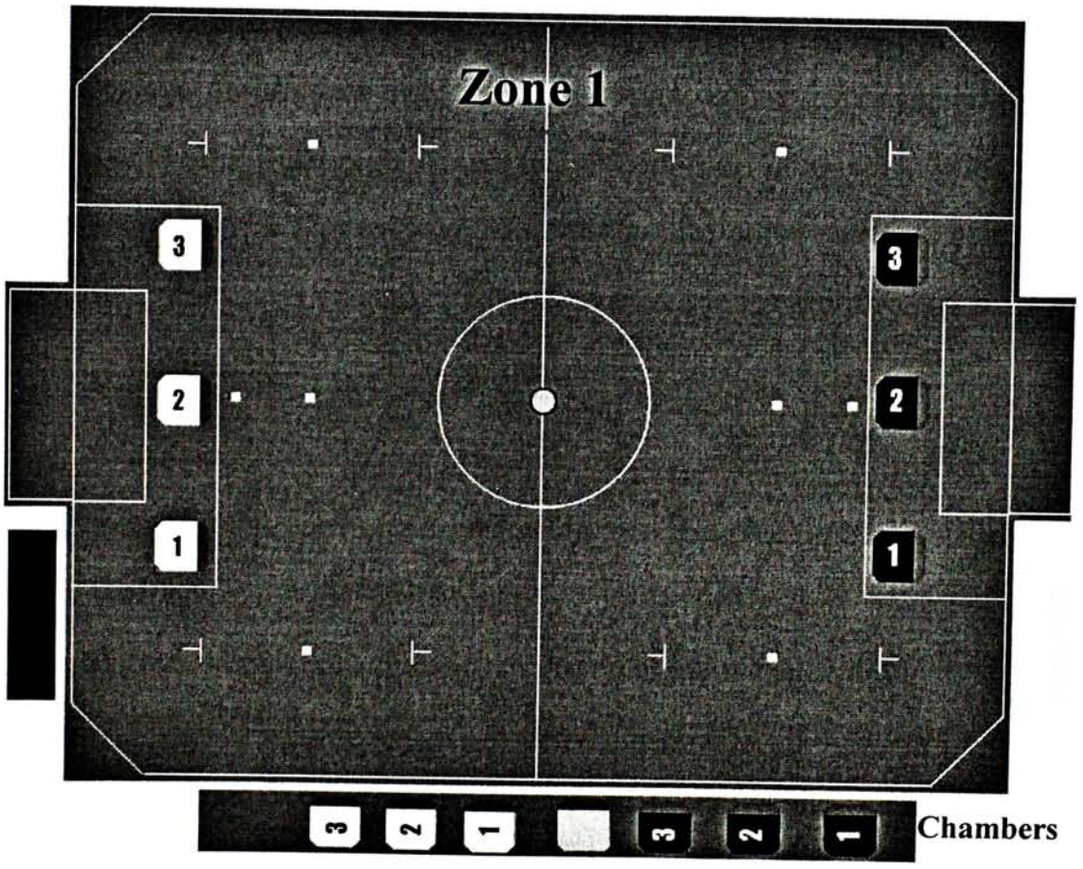


Figure 4.16 Initial positions of the robots

Chapter 5 Auto-Scoring and Auto-Judging

Scoring and judging take a high proportion in maintenance workload. Previously, judges had to judge whether there is a goal. The game must be stopped if either side goals and then ball and robots must be repositioned. All the tasks above were done manually. It is truly a heavy workload.

Actually, auto-scoring mechanism is comparatively easier to be realized. We only have to monitor the coordinate of the ball. Once the server found that the coordinate of the ball is located within the goal areas, we add one point to the team who scored. Of course there are some details to be considered which is to be discussed later.

Auto-judging is a bit more complicated. We have to build an expert system in order to establish a mapping from the rules to the corresponding actions to take. Once the system detected that some rule is violated, the system is altered immediately to perform the corresponding actions such as goal kick, penalty kick, etc.

5.1 Auto-scoring

When the ball entered either goal, add a point to the corresponding team. The color bars below the game field video window indicate the sides. Take the case in Figure 4.12 for example. It means that the left half-court belongs to team blue while the other half belongs to team yellow. So, in this case, if the ball entered the goal area of blue (ball.x<0), no matter which side kicks it last, we add a point on the score of team yellow; and vice versa (ball.x >220 indicates the goal of the other side).

Once the ball enters either goal area, the game is paused immediately to avoid repeated scoring. Then the judge robot is assigned to reposition the ball to the center of the game field. The judge robot is then driven into the charging chamber automatically with APF.

5.2 Auto-judging

There are two main components in auto-judging mechanism: the mapping from the rules to the corresponding actions and the judge robot. The mapping is to determine which rule is violated and what action is to be performed. We established an expert system for this. The expert system focuses on the rules of IRIS. It will give the solutions (the actions to be taken) according to the status of the game when there are violations to the rules. Such actions are usually repositioning of robots and ball. We can move the robots to the desired positions with the help of the motion controllers introduced in chapter 4. However, we cannot drive the ball by itself. We therefore designed another kind of robot for positioning the ball: the judge robot.

The expert system

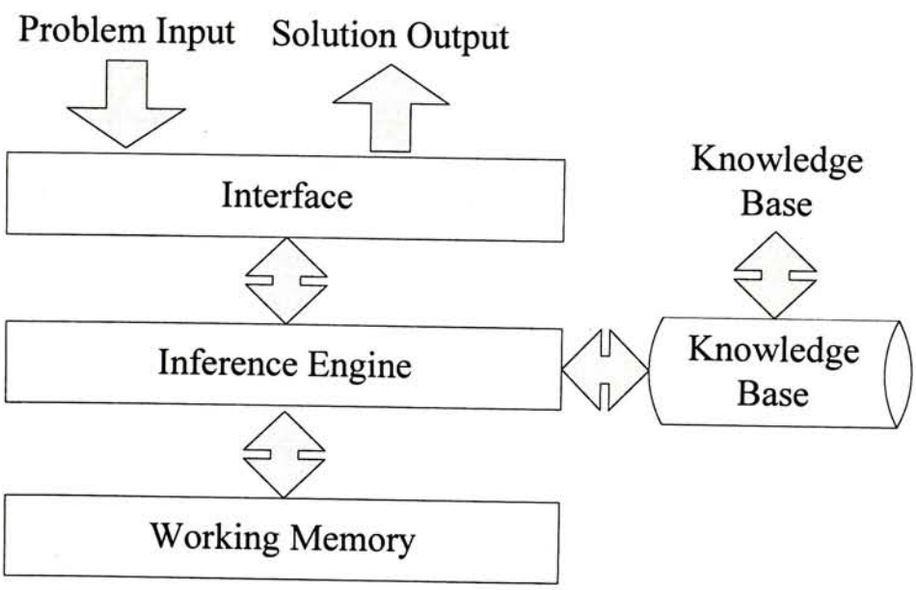


Figure 5.1 Architecture of a typical expert system

The flowchart above is the architecture of a typical expert system. There is a knowledge base in which we can define the rules. The rules are typically of *if...then...* form:

If	*Then*
All three robot of the same team stay inside their own GK for over 3 seconds.	Free Kick for the other side
More than one robot of the same team stay inside their own SG for over 3 seconds	Penalty Kick for the other side
More than two attacking robots of the same team stay inside the SG of the other team for over 3 seconds	Goal kick for the defense side
The ball stays inside the SG of the defense side for more than 10 seconds but no goal	

Table 5.1 Typical rules in IRIS

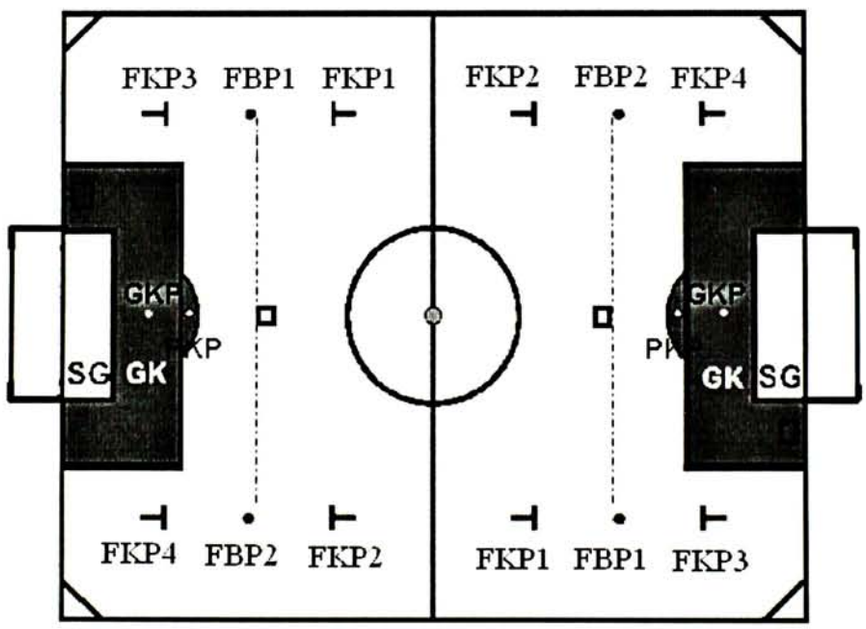


Figure 5.2 Particular zones and points on the game field

where SG: goal area; GK: Penalty area; FKP: Free kick point; PKP: Penalty kick point; GKP: Goal kick point; FBP: Face-off point.

Free kick is to position the ball at the nearest free kick point (FKP) away from the foul position. Then the goal keeper is located inside the SG of its own side. The other two robots of the defense side must leave at least 450 mm away from the ball. Here, for simplicity, we drive them to the center circle. Robot position of the attacking side

is not restricted. Both sides are not allowed to move the robots until they see 'Start' in the status bar of the flash interface.

Penalty kick is to put the ball at the nearest penalty kick point (PKP) with the offenders' side. Goal keeper of this side is also located at point 1 inside its own SG. Robot number 1 of the attacking side takes the kick. It is located at point 2. All robots other than those two above are required to leave at least 300mm away from point 2. Here, for simplicity, we also drive them to the center circle. When the 'Start' is displayed, all robots can move without any restriction.

During goal kicks, only the goal keeper is allowed inside the GK of the kicking side. The ball is positioned at the goal kick point (GKP). All robots of the opposite team must keep a distance of at least 450mm away from the GKP. We drive them to FKP1 and FKP2 of their own half (the third one is the goal keeper and is always inside its own GK). Then, drive the other 2 robots to FKP1 and FKP2 on the kicking side half court.

When there is a goal, game is paused immediately. Then, the ball is repositioned at the center of the game field by judge robot. Other robot players are driven back to their initial positions. Robot 1 of the team who has just lost a point can be positioned closer with the ball (near the center circle). After that, game resumes.

Those rules above are only a part of rules defined in the datasheet of IRIS. However some rules are not easy to be judged in automated cases. Take 'deliberate pushing' for example, such word is so abstract that it is not easy to judge it accurately without human interference. This would need more future works. However, for the comparatively simpler cases as listed in Table 5.1, the expert system would have a good performance.

5.3 Judge robot

We mentioned about positioning of the ball which requires the help of the judge robot. The judge robot is built on the basis of the original robot soccer by adding a circle

gripper in front. The gripper is controlled by setting the value of [i*4+4] in table 4.6. For judge robots we have i=3 (team 0) or i=4 (team 1).

[i*4+4] value	Action
0x00	Stay
0x02	Lower down the gripper
0x08	Lift the gripper

Table 5.2 A byte controlling the gripper of robot soccer

Thus the judge robot can be controlled to catch the ball by lower down the circle when approaching it and release the ball when it gets to the desired point. After the judge robots finished their task, they are driven back to the charging chamber. After that, the game is resumed.

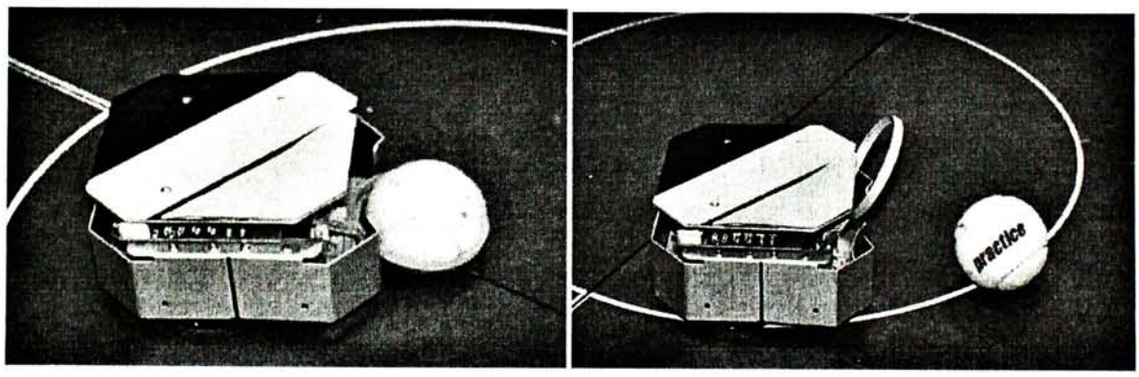


Figure 5.3 Judge robot catches and releases the ball

We added a new robot member in our system: judge robot. In order to distinguish it from other robots, we added a new color pattern for it:

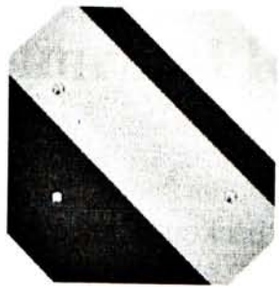


Figure 5.4 Color patches of judge robot

The team patch of the judge robot is blue, while the member patch is newly added: grey. Because of this, we have to modify the vision part to identify this new robot.

When judge robot is on the game field, including 6 robot players, there are totally 7 robots in the vision. Firstly we divide them into two teams: blue and yellow. This step is similar with the previous 6 robots case: We find 7 largest patches (team patches) and then sort them according to the average Cb value. Top 4 patches with the largest Cb value are blue patches, and the rest three are yellow ones. Here, judge robot is classified into team blue. Then we sort these robots according to the Cr values their member patches. Team yellow case is absolutely the same with the previous approach: the one with the highest Cr value is the red patch, which is number one. The second largest value indicates the blue patch, number 3. The patch with the smallest Cr value is the yellow patch which is number 2.

Team blue case is a bit different since we added a new member in it. After many experiments, we found an algorithm to identify the members of team blue including the judge robot. First steps are the same: sort according to the Cr values of their member patches. Red has relatively largest Cr value while blue has the lowest. We can therefore easily find number 1 and number 3 of team blue. For the rest two, in our experiments, their Cr value is so near. It is very likely to make mistakes if we simply use Cr values to judge. Here, we compare the average Cb values of their member patches as well. The color with a larger Cb value is grey. The corresponding robot is the judge robot. The other with the lower Cb is yellow which indicates robot number 2.

In our design, there are 2 judge robots in each game. One of them is used as the

chief judge and the other one is for backup. They exchanges automatically once the chief judge runs out of power. The chief judge is assigned to chamber 8 where there is no charging plug. Because if there is a charging plug, when there is a need of judge robot, it will cost several seconds to unplug. For efficiency, we just need the judge robot appear as soon as possible. Consequently, we cancelled the charging mechanisms of the chief judge. Meanwhile, the backup judge robot is charging at chamber 4.

Chapter 6 Experimental Results

As above, we can construct an internet based automated robot soccer system. Here, in order to evaluate the performance of it, we designed four cases to test:

1. Change one robot
2. Change all six robots
3. Actions after a scoring
4. Actions after a foul

1 and 2 are testing Auto-judging while 3 and 4 are for Auto-Judging and Auto-Scoring Mechanisms. Before all testing, we always reset the chamber entrance coordinates as introduced in Chapter 4.5 in order to lower down the failure rate. After the calibration, we can start our test.

First, we see the performance of changing one single robot. (a) is the original positions of robots. Then, we set ACS manually by setting 'ROBOT' to 'Blue 1' and 'THRSD' to 'Always' (Figure 4.12). It means that we change robot Blue 1 without regarding its battery level. Then, robots on the game field are driven to their original positions by APF (b). When they stop, robot Blue 1 is masked on the video input, then door of chamber one is opened and the fully charged Blue 1 goes out of the chamber to Zone 1 in Figure 4.15 using APF (c). Then, the new Blue 1 is masked instead, and the low-power counterpart reached to the midpoint (point 2) (d). Then, it continues to move under the control of CRC until it enters chamber 1. Door of chamber 1 is closed and charging starts automatically (e). Lastly, in (f), all robots on the game field are driven to the original positions as (a).

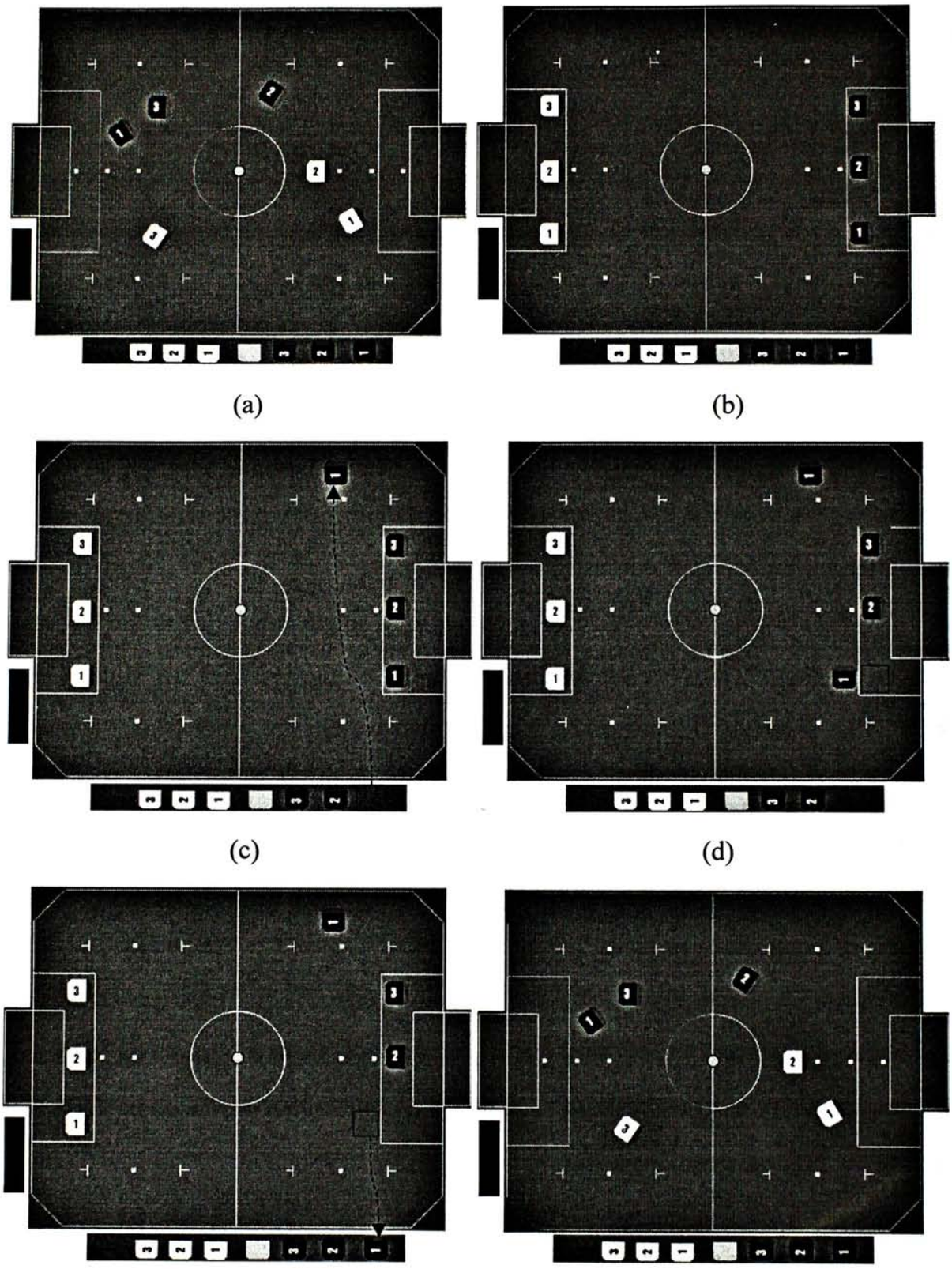
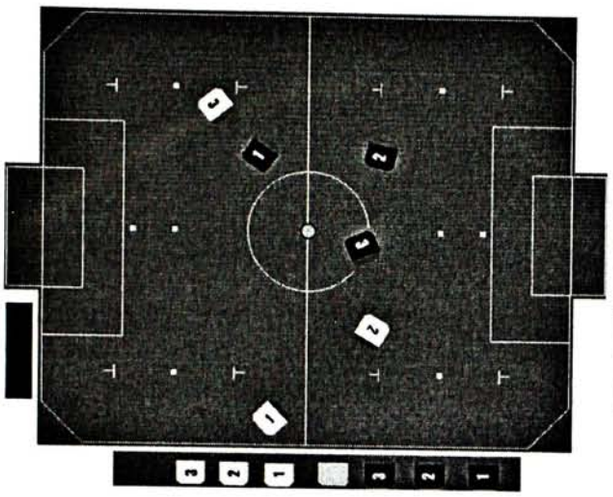


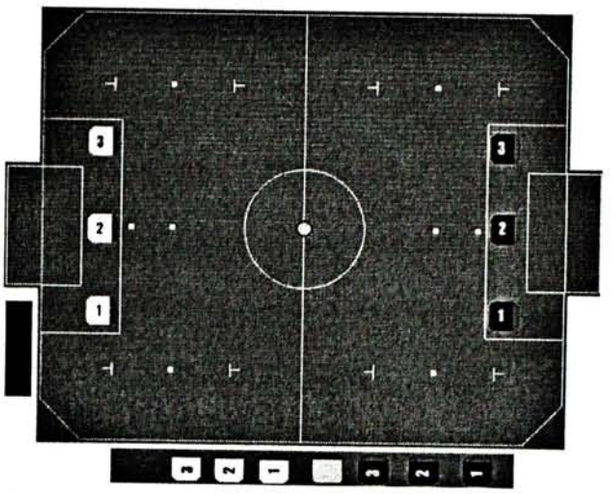
Figure 6.1 Procedure of changing one robot

The next issue is to change all 6 robots. We do this whenever a new game is started. Firstly we drive all six robots from the original positions (a) to initial

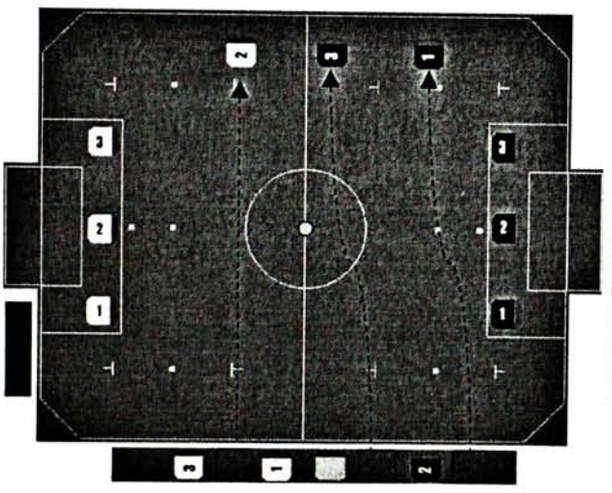
positions (b). Robots of group 0 on the field are masked and the chamber doors of three robots of group 0 (Table 4.11) are opened. Those three robots are driven to zone 1(c). Then, mask the three robots instead and drive low-power robots to their midpoints (d) and enter the chambers (e). Similarly we can change the robots of group 1 (f)~(h). Finally, all robot is driven to their initial positions (b) for a new contest.



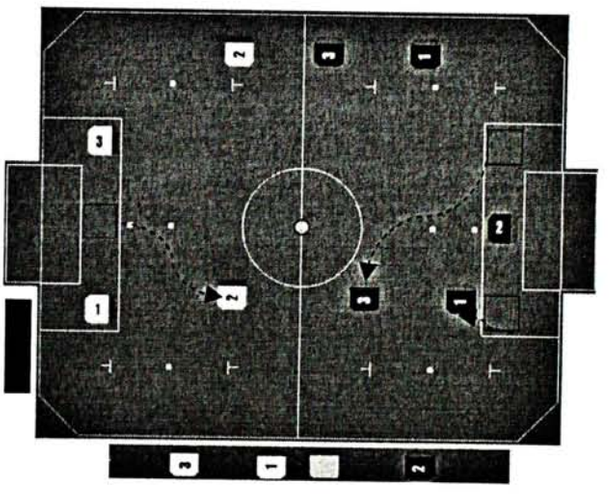
(a)



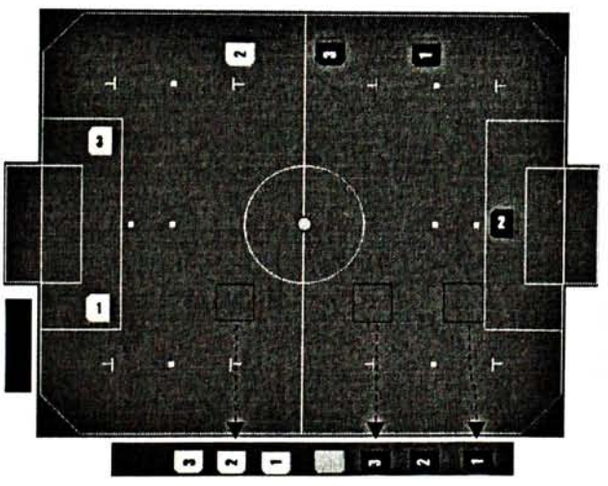
(b)



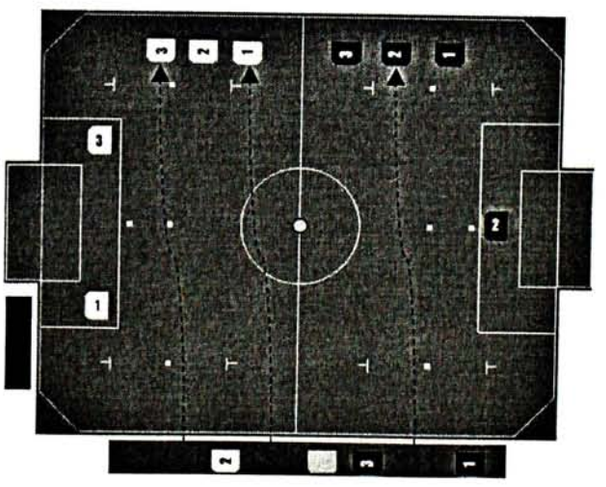
(c)



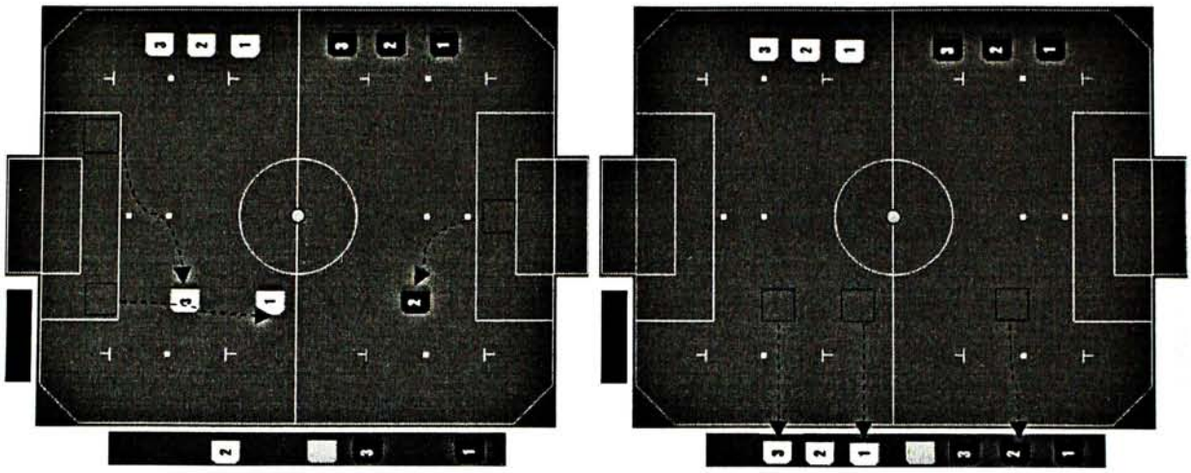
(d)



(e)



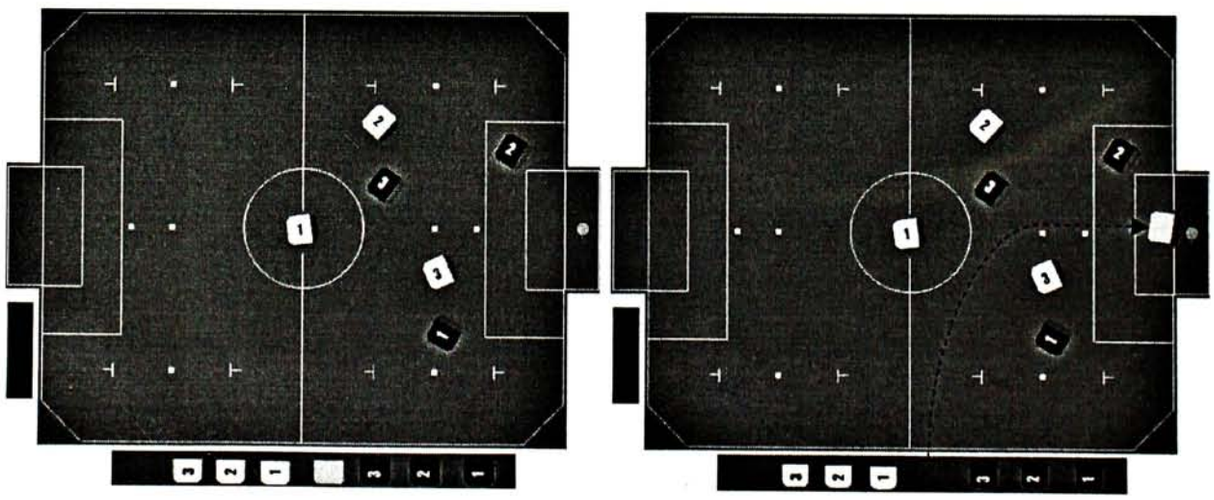
(f)



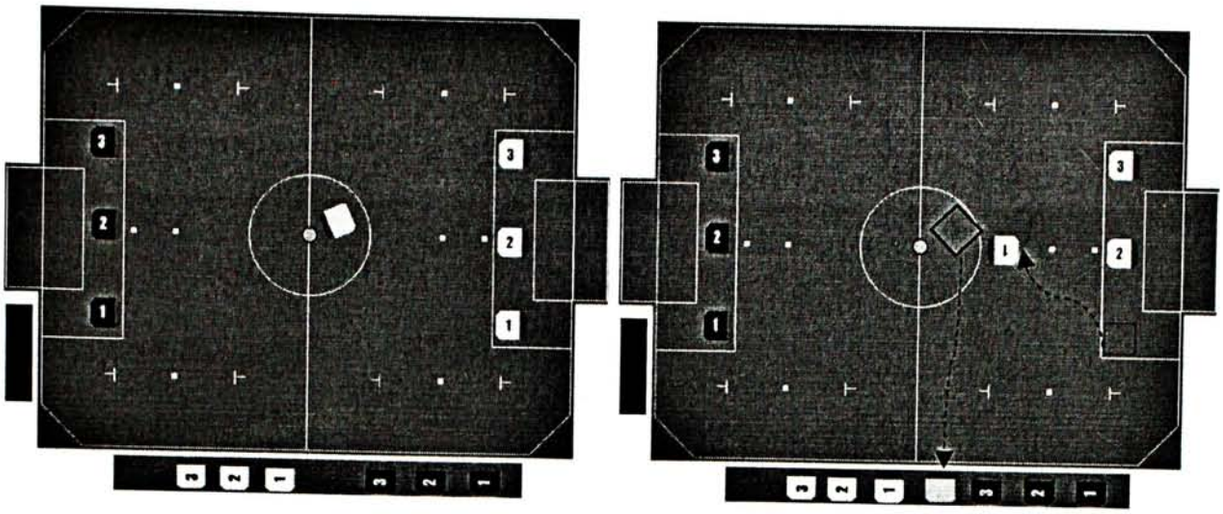
(g) (h)

Figure 6.2 Procedure of changing six robot

Now it is time to test our new member: the judge robot. Here we control blue 2 to shoot into the goal of yellow. After the ball entered the goal area of yellow (ball.x>220), the game is paused immediately (a). Then, judge robot is driven out of its chamber to a adjacent of the ball using APF control (b). After this, it lowers down the gripper to fetch the ball and bring it to the center of the game field. Other robots are driven back to their initial positions (c). Finally, judge robot releases the ball and goes back to its chamber. The procedure for this step is similar with the one used in single robot changing. Robot 1 of team yellow is moved to a point near the center circle to launch the ball (d).



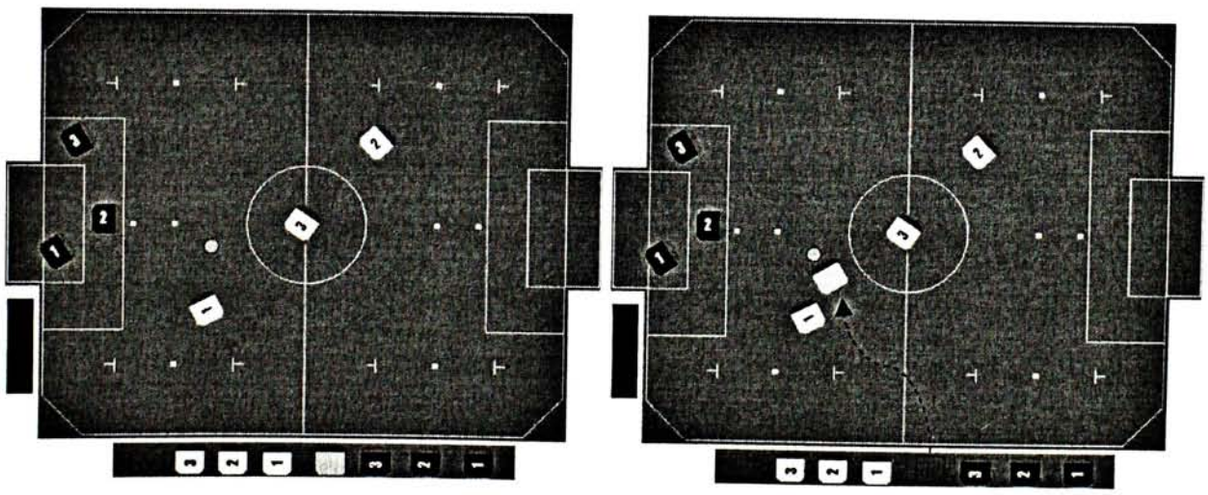
(a) (b)



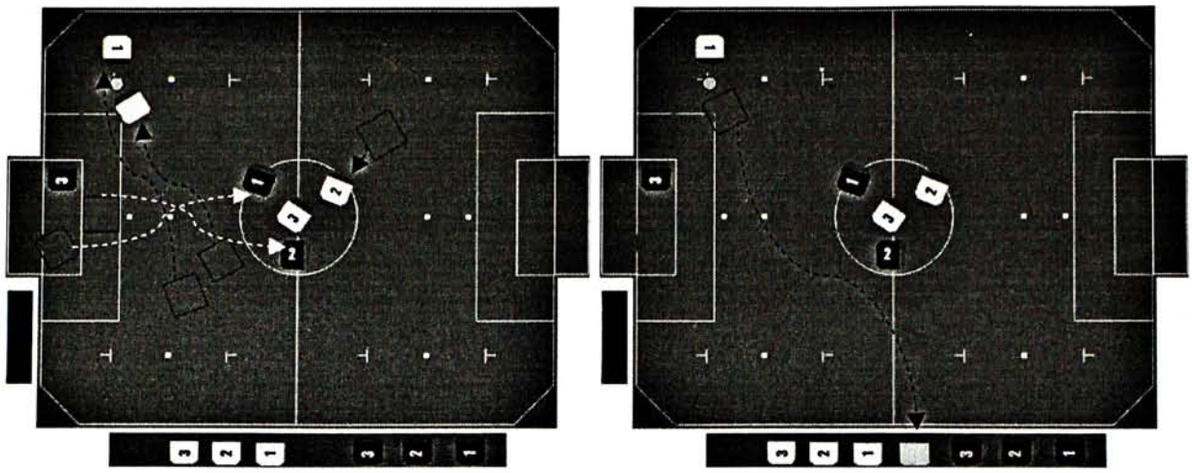
(c) (d)

Figure 6.3 Processing procedure after a goal

Lastly, we have to examine the auto-judging mechanism. We put robot blue 1 to 3 inside their own GK (a). After three seconds, the condition of the first rule of Table 5.1 is satisfied. Expert system of auto-judging says that it is time for yellow team to take a free kick. Then, judge robot is driven to take the ball to FKP3 (b) (c). Number 1 of team yellow is driven to a point near FKP3 facing the goal. All other robots except robot 3 of blue (the goalkeeper) are driven to the center circle.



(a) (b)



(c) (d)

Figure 6.4 Processing procedure after a goal

Chapter 7 Conclusions

7.1 Summary

In this thesis, we have proposed and verified a new automatic internet-based robot soccer system which can conduct automatic games without human intervention. Adaptive motion controllers (APF, CRC) are developed to drive robots to a desired position, where APF is used when obstacle avoidance is needed. CRC is used where no obstacle is needed to be considered and high precision is required. Those controllers are widely used in auto-charging and auto-judging mechanisms. The performance is fast enough and the accuracy is satisfactory.

An expert system which describes the game rules is established. It monitors the positions of the robots and the ball and sees whether it is necessary to take actions. The RF communication part and vision part have been redesigned on the basis of the existing IRIS system. Flash programs are published and the server program is rewritten in order to meet the requirement of the new system.

In aspect of hardware, a judge robot is also designed. Equipped with a gripper, it can help move the ball. Meanwhile, a new game field with charging chambers is also made.

Now, players from all over the world can log on to their accounts and play the game remotely. Server will control the game course instead of human judges, including start, time-out, fouls and scorings etc. The previously boring tasks such as moving the robots now are done by the robots themselves. Reposition of ball is also done by the new designed judge robot. The competition information and the scores are managed by the web server. Finally, the performance of the new system is validated by simulations and field experiments.

7.2 Future work

The system just we developed is not stable enough. There are several issues we should do further:

We used the basic APF in this first version. As mentioned above, it has a problem of local minima. This is because that only the local information is taken in to consideration, the shape of potential field after the superposition is unpredictable. We can make use of those improved APF approaches, such as [18], [20]. They can avoid local minima and have a better convergence.

Vision part should also be improved. The existing vision system is not very robust under various lightings. Sometimes it fails to identify robots if the parameters are not well set. Here, [8] presents a system that can adapt to rapidly varying light and coloring conditions, while maintaining speed and accuracy. [30] introduced a vision system that does not need any calibration and adapts to changing lighting conditions during run time. We may take them for references in the future to improve the performance of the vision part. [31] presented an exemplary robot soccer vision system that can be easily extended to multi camera case which is perfectly suitable for our IRIS.

Because of the narrow entrance of chambers and uneven floor, entrance failures happen from time to time. The problem is that the computer vision is not able to monitor the situations of the chamber area. It can only cover the game field area. We have to add another camera for this area [31], or we can choose another controller when the robot is entering the chambers. For example, we can use line tracing at the stage of entering the chambers, because the lines are comparatively stable with respect to the game field. It will bring a higher successful rate.

Currently we move the ball with the help of the judge robots. In the future, we may let the ball move by itself by design it as a spherical robot [32]. Then it can roll to any desired position without any external propelling force. The judge robots are no need to be used either.

Add new rules into the knowledge base of the expert system. Make the

auto-judging mechanism more accurate and intelligent.

Auto-play mechanism should be considered in future. Players can upload their own program to play rather than the current manual control. IRIS will be more attractive and educational by then.

Appendix A

Playground Dimensions of IRIS

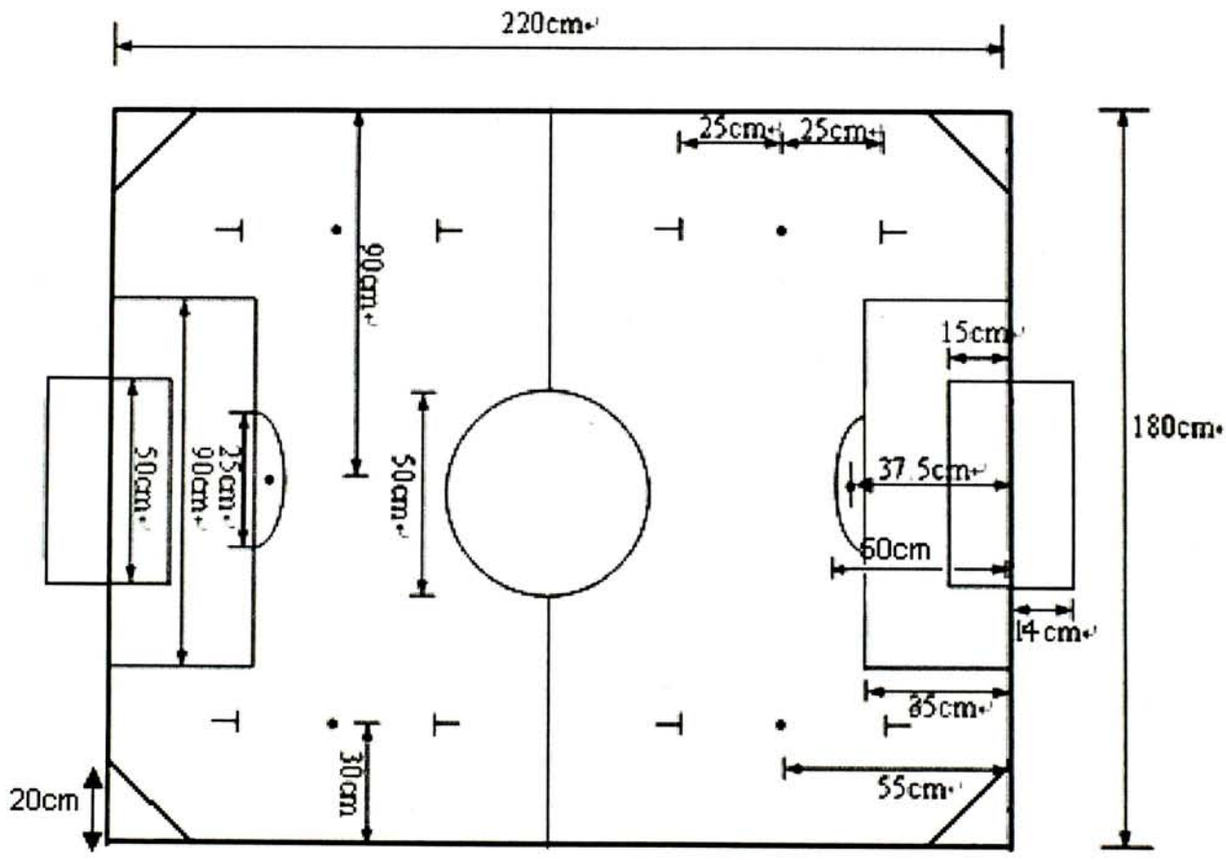


Figure A.1: The playground dimensions of IRIS (source: IRIS Datasheet)

Bibliography

[1] J. van der Linden. Dynamic avoidance control of soccer playing mini-robots. Master's thesis, University of Twente, 2005.

[2] FIRA MiroSot Game Rules For Middle League and Large League [M].

[3] G. Mester, Motion Control of Wheeled Mobile Robots, 4th Serbian-Hungarian Joint Symposiumon Intelligent Systems, SISY, 119-130, (2006).

[4] R. Fierro and F. L. Lewis, "Control of a non-holonomic mobile robot using neural networks," IEEETrans. on Neural Networks, vol. 9, no. 4, pp. 589–600,1998.

[5] W.D.J. Dierssen."Motion planning in a robot soccer system. Master's thesis, University of Twente, 2003.

[6] M. Vandittelli G. Oriolo, A. Luca. Dynamic Feedback Linearization: Design, Implementation and Experimental Validation. IEEE Transactions on Control Systems Technology, Secaucus, NJ, USA, 2002.

[7] J.P. Laumond. Robot Motion Planning and Control. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[8] Wyeth, G. and Brown, B., 2000. "Robust Adaptive Vision for Robot-Soccer". Mechatronics and Machine Vision in Practice.

[9] Weiss, Norman; Hildebrand, Lars: "An Exemplary Robot Soccer Vision System". In: P. Kopacek et al. (Eds.): CLAWAR / EURON / IARP Workshop on Robots in Entertainment, Leisure and Hobby ELH'04 Proceedings, Technische Universität

[10] Gregor Klancar, Omar Orqueda, Drago Matko, Rihard Karba, "Robust and Efficient Vision System for mobile robots control-application to soccer robot", Elektrotehni¡ski vestnik 68(5): 247–253, 2001, Electrotechnical Review, Ljubljana,Slovenija

[11] Vieira, F. C., Alsina, P. J., and Medeiros, A. A. D. (2001). Micro-robot soccer team - mechanical and hardware implementation. In Congresso Brasileiro de

Engenharia Mecˆanica, pages 534–540.

[12] G. Dudek and M. Jenkins, Computational Principles of Mobile Robotics (Cambridge University Press, 2000).

[13] Gregory Dudek and Michael Jenkin. Computational Principles of Mobile Robotics. Cambridge University Press, Cambridge, UK, 2000.

[14] R. Seesink, W. Dierssen, N. Kooij, A. L. Schoute, M. Poel, E. Schepers, and T. Verschoor. Fast data sharing within a distributed multithreaded control framework for robot teams. In J. F. Broenink, H. W. Roebbers, J. P. E. Sunter, P. H. Welch, and D. C. Wood, editors, Communicating Process Architectures 2005 (WoTUG-28), volume Concurrent Systems Engineering Series 63, pages 147–154, Eindhoven, The Netherlands, Sep 2005. IOS Press, Amsterdam.

[15] N.S. Kooij. The development of a vision system for robotic soccer. Master's thesis, University of Twente, 2003.

[16] R. D'Andrea, T. Kalmr-Nagy, P. Ganguly, and M. Babish. The Cornell Robocup Team. In RoboCup 2000: Robot Soccer World Cup IV, pages 41–51, 2001.

[17] Sørensen, Mathias (2003). Artificial potential field approach to path tracking for a non-holonomic mo-bile robot. In: Proceedings of the 11th Mediter-anean Conference on Control And Automation.

[18] Ge, S. S. and Cui, Y. J., "Dynamic Motion Planning for Mobile Robots Using Potential Field Method," Autonomous Robots, vol. 13, pp. 207-222, 2002.

[19] Khatib, O., "Real-Time Obstacle Avoidance for Manipulators and Mobile Robots," International Journal of Robotic Research, vol. 5, pp. 90-98, 1986.

[20] Kim, J.-O. and Khosla, P. K., "Real-Time Obstacle Avoidance Using Harmonic Potential Functions," IEEE Transactions on Robotics and Automation, vol. 8, pp. 338-349, 1992.

[21] Leonard, N. E. and Fiorelli, E., "Virtual Leaders, Artificial Potentials and Coordinated Control of Groups," in Proceedings of IEEE Conference on Decision and Control, 2001, pp. 2968-2973.

[22] Rimon, E. and Koditschek, D. E., "Exact Robot Navigation Using Artificial Potential Functions," IEEE Transactions on Robotics and Automation, vol. 8, pp. 501-518, 1992.

[23] Michael A. Goodrich, "Potential Fields Tutorial", 2002.

[24] Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control," in Proceedings of ASME Conference of Robotic Research: The Next Five Years and Beyond, 1984.

[25] Volpe, R. and Khosla, P., "Manipulator Control with Superquadric Artificial Potential Functions: Theory and Experiments," IEEE Transactions on Systems, Man and Cybernetics, vol. 20, pp. 1423-1436,

[26] Ge, S. S. and Cui, Y. J., "New Potential Functions for Mobile Robot Path Planning", IEEE Transactions on Robotics and Automation, vol. 16, pp. 615-620, 2002.

[27] Connolly, C. I., "Applications of Harmonic Functions to Robotics," in Proceedings of IEEE International Symposium on Interlligent Control, 1992, pp. 498-502.

[28] Leng-Feng Lee and Venkat Krovi, "A Standardized Testing-Ground for Artificial Potential-Field based Motion Planning for Robot Collectives", 2006.

[29] Manuela Veloso, Michael Bowling, Sorin Achim, Kwun Han, and Peter Stone. The CMUnited-98 champion small robot team. In RoboCup-98: Robot Soccer World Cup II. Springer Verlag, 1999.

[30] M. J¨ungel, J. Hoffmann, and M. L¨otzsch, "A real-time auto-adjusting vision system for robotic soccer," in 7th International Workshop on RoboCup 2003 (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence, (Padova, Italy), Springer, 2004.

[31] Weiss, Norman; Hildebrand, Lars: "An Exemplary Robot Soccer Vision System". In: P. Kopacek et al. (Eds.): CLAWAR / EURON / IARP Workshop on Robots in Entertainment, Leisure and Hobby ELH'04 Proceedings, Technische Universität

[32] S. Bhattacharya and S. Agrawal, "Spherical Rolling Robot: Design and

Motion Planning Studies", IEEE Transactions on Robotics and Automation, 16(6): 835-839, 2000.